1-2/1989

KLUB MIKROELEKTRONIKY

ATARI®

ZPRAVODAJ

OLOMOUC

Vážení zájemci o výpočetní techniku, především o Atari

Redakce Vás vítá již do třetího ročníku Olomouckého zpravodaje. Zájem o zpravodaj plně potvrzuje hlavní myšlenku, proč jsme před dvěma roky začali s přípravou a vydáváním. Co nejkratší a nejrychlejší cestou dostat mezi uživatele výpočetní techniky informace a popisy o stávajícím softwarovémm a hardwarovém vybavení 8-bitových počítačů Atari a hlavně jejich využití v praxi. Nejde nám o to vytvořit odborný zpravodaj specializovaný na určitou oblast, ale snažíme se zařazovat takové materiály, aby byl zajímavý nejen pro pokročilé, ale přístupný a srozumitelný i pro začátečníky.

Interní informace, týkající se vlastního života ZO Svazarmu - Atari Klub Olomouc, jsme do zpravodaje nezařazovali vzhledem k časovému rozdílu od vlastní přípravy zpravodaje do tisku, až po skutečné vytištění a distribuci. Mohlo by se klidně stát, že to, co bychom inzerovali jako novinku, by v době expedice zpravodaje již bylo zastaralé. Ze stejných důvodů nezařazujeme ani inzeráty o prodeji a koupi. Máme zjištěno, že v mnohých případech se inzerované informace nezakládají na pravdě již v době, kdy přijdou poštou na náš stůl. Pokud se týká ostatní inzerce, rádi vyhradíme místo pro informace o vznikajících skupinách s konkrétním zaměřením, tak jak jsme to v průběhu uplynulého ročníku inzerovali a postupně tiskli.

Co dál?

Uvažovali jsme, jestli dál vydávat náš zpravodaj. Po srovnání "PRO" a "PROTI" vydávání olomouckého zpravodaje zvítězil názor pokračovat v další publikační činnosti. Chceme Vás informovat nejen o programech, ale také o novinkách 8-bitových počítačů Atari, které se neustále objevují, i když někteří považují tuto výpočetní techniku za překonanou.

V rozsahu a náplni bychom chtěli dodržet obdobnou strukturu jako dosud. Chtěli bychom se v každém zpravodaji vracet k jednomu z nejrozšířenějších programovacích jazyků na světě - jazyku PASCAL, pro který chceme vyhradit pár stránek v každém čísle, příštím číslem počínaje. Nepůjde nám o vlastní výuku, ale o zkušenosti z Vašich řad, doplněné praktickými programy (nejlépe pro jazyk KYAN PASCAL). Další pravidelné specializované rubriky zavádět nebudeme. Mohlo by se stát, že bychom je potom museli naplnit na úkor zajímavějších materiálů a novinek.

Pokud budou stejné hospodářské a finanční podmínky jako v uplynulém období, plánujeme opět vydání 8 čísel zpravodaje za rok za stejný klubový příspěvek 30 Kčs. Znovu upozorňuji, že není v silách a možnostech naší organizace provádět distribuci jednotlivcům, ale pouze klubům na základě vzájemné dohody, respektive kolektivního členství v naší ZO Svazarmu.

Na závěr chci připomenout, že zpravodaj nevydáváme pro naši větší slávu, ale pro větší informovanost všech. Vyzýváme ty z Vás, kteří jste schopni a ochotni přispět do zpravodaje, ke spolupráci. Zasílejte své příspěvky na naši adresu. Honorář máme možnost vyplácet v rámci směrnice pro hospodaření ZO Svazarmu.

ing.Pavlík Dobromil
ZO Svazarmu AK Olomouc

Kyan PASCAL
Kyan Software, Inc.
1850 Union Street #183
San Francisco, CA 94123
U.S.A.

Recenze programu
KYAN PASCAL

Bob Curtin

Začal jsem si hrát s počítači poměrně pozdě a pro mě, stejně jako pro většinu vlastníků počítačů Atari, byl BASIC mým prvním programovacím jazykem, který jsem se učil. Musím říci, že je to jazyk, ve kterém dosud dělám většinu svých programů. BASIC je jednoduchý a snadný na výuku a používání, je výkonný a interaktivní. Ale je také pomalý, náročný na paměť a smutně proslulý jako propagátor špatných návyků při programování.

Časy se mění. Větší náročnost na paměť dnes není žádný problém. Většina domácích počítačů má 64 KB RAM, některé dokonce mnohem více.

Ale pomalost, to je jiná. Znamená mnoho, když používáme P/M grafiku, vektory přerušení nebo plovoucí řádovou čárku. Ano, vím, některé strojové rutiny pracují velmi dobře v BASICu a dokáží podstatně zrychlit běh programu, ale většinou máme problémy s implementací.

Co teď? Programovat v jazyku C? Výborná myšlenka, ale musíte mít disketovou jednotku. Jazyk C je rychlý, kompaktní, srovnatelný s assemblerem a jednoduchý k naučení. Podle mého názoru, je to vůbec nejlepší jazyk pro mnoho oblastí, a to právě svou snadnou přenosností z jednoho typu počítače na druhý. V současnosti je na 8-bitovém počítačovém trhu vysoce hodnocena verze jazyka C pro počítače Atari pod názvem DEEP BLUE C.

Naštěstí firma Kyan Soft přišla na trh s jazykem KYAN PASCAL (kompilátor/editor jazyka PASCAL), který mě dokonale uspokojil. Tento PASCAL je podobný jazyku C. Vždyť PASCAL patří mezi strukturované jazyky - je to jeden člen z rodiny "ALGOLů", kam patří i PL/C, C nebo Ada.

Co je to strukturované programování? Abych vám řekl pravdu, nechci se vrtat v těžké a přesné definici. Pouze vám sdělím, že jde o disciplinované přistoupení k problému programování na počítačích (Většina programů v BASICu je tvořena tzv. systémem "shooting from the hip", což česky znamená "střelba od boku".).

Strukturovaný PASCAL vyžaduje více systematického přístupu k programátorskému umění. Výsledkem je program nekonečně větší jasnosti a výkonnosti. Faktem zůstává, že jazyk PASCAL byl původně vytvořen pro výuku systematické techniky programování

Co říci o rychlosti? PASCAL je zkompilovaný jazyk, který pracuje průměrně 5 krát rychleji než BASIC.

Ve zdrojovém kódu PASCALu mohou být také použity rutiny v assembleru (strojový kód 6502). Kompilátor jednoduše přeloží strojový jazyk, když ho najde. Vytvořil jsem si knihovnu procedur napsaných v PASCALu a assembleru, které mohu použít znovu a znovu ve svých dalších programech.

Například PASCAL nemá funkci náhodného generování čísel, tak jsem napsal krátký podprogram ve strojovém jazyku na získání náhodných čísel z adresy $D20A a upravil ho tak, aby fungoval bez omezení. Definoval jsem podprogram jako proceduru, která může být kdykoli volána z hlavního programu.

Kombinace PASCALu a assembleru je opravdu neuvěřitelně výkonná. Zajišťuje jednoduchost typu vyššího programovacího jazyka a přitom uživateli dává rychlost a řízení "čistého" assembleru, kdykoli je třeba. Tuto kombinaci je nejlepší využít v dlouhých programech náročných na čas, kde brzy oceníte jazyk assembler, který čas získává zpět.

Další výhodou PASCALu je jeho přenosnost. PASCAL většinou můžeme zkompilovat na odlišných počítačích prakticky beze změny zdrojového kódu. To je opravdová výhoda pro ty, kteří programují nebo obsluhují několik různých počítačů.

Toť vše o vhodnosti jazyku PASCAL. A co řekneme o KYAN PASCALu? Když mi předávali soupravu jazyka KYAN PASCAL ke zhodnocení, nejvíce mne zajímalo, zda je KYAN PASCAL plnohodnotná verze jazyka PASCAL nebo pouze limitovaná sada programů. Teď je to definitivně rozhodnuto! KYAN PASCAL je výkonná implementace jazyka PASCAL a obsahuje navíc několik procedur (Plot, Drawto, Position atd.), abychom mohli plně využít jednoduchosti a výhod grafické kapacity 8-bitových počítačů Atari.

Souprava obsahuje buď disk nebo msf. kazetu (standart i TURBO) s kompilátorem, editorem, knihovnou funkcí (LIBRARY), mnoha procedurami a několika demonstračními programy (tyto programy jsou jak ve formě PASCALu, tak v zkompilované verzi - pouze na disku).

Dokumentace je ve formě dobře napsaného a zhuštěného manuálu s pomocným indexem. Uživatelský manuál, kombinovaný s učebnicí jazyka PASCAL, nám přináší možnost rychlého proniknutí do problematiky programování v jazyku PASCAL. (U nás zatím vyšly dva tituly, které vznikly překladem a výběrem kapitol z tohoto manuálu. První z nich, stručnější a určený pro pokročilejší programátory, vydal Atari Klub v Plzni, druhý, obsáhlejší a určený i pro začátečníky, vydala ZO výpočetní techniky Svazarm Hodonín. Obě publikace mají název Kyan Pascal. - poznámka překladatele)

Abychom mohli spustit kompilátor, potřebujeme minimálně 64 KB RAM. K dispozici je však nyní i 48 KB verze kompilátoru, která vznikla později.

Testoval jsem verzi KYAN PASCALu pro počítač Atari 800 XL/XE i verzi tohoto jazyka pro počítač Atari 130 XE (obsahuje RAMDISK) . Editor a kompilátor obou verzí jazyka KYAN PASCAL pracovaly ke vší spokojenosti.

Ale rozdíl je neuvěřitelný. Trvá přes 45 sekund, než

"natáhnu" kompilátor z disku do paměti, ale když použiji RAMDISK počítače Atari 130 XE, doba potřebná na nahrátí kompilátoru se sníží na necelých 2.5 sekundy. Kolik tak ušetří programátor času! Odstraňování chyb ve zkompilovaném programu je časově velmi náročné, programátor přeskakuje mezi editorem a kompilátorem do DOSu a zpět. Použití RAMDISKU u počítače Atari 130 XE vše vyřešilo.

Editor KYAN PASCALu využívá všechny možnosti editace pomocí řízení kurzoru jako Atari BASIC. Další funkce editace, jako je vyhledávání a změny, manipulace se soubory nebo vsunutí souboru do programu, jsou přístupné kombinací CONTROL+(klávesa). Funkce Editoru tedy ovládáte kombinací jedné nebo dvou kláves, což přispívá ke zvýšení komfortu programování. DOS je přístupný na jednoduchý příkaz a s jeho pomocí můžeme přehrávat Editor nebo Kompilátor do paměti (pomocí volby L DOSu - binary load).

KYAN PASCAL používá standartní diskový operační systém - Atari DOS 2.5. Pokud vlastníte disketovou jednotku Atari 1050, můžete použít zhuštěnou formu zápisu na disk. Pro majitele počítačů Atari 130 XE obsahuje volbu RAMDISKU k podstatnému zrychlení času kompilace programu a překladu assembleru.

Kompilátor jazyku KYAN PASCAL sice pracuje pomalu, ale vzhledem k jeho účinnosti je tato pomalost pravým požehnáním. Kompilátor má totiž plně automatické rutiny ke sledování chyb a indikátory syntaktických chyb. Také odhaluje některé zmetkové chyby a "odchytá" přes polovinu chyb v assembleru. Když kompilujete nový program v PASCALu, na obrazovce průběžně vidíte výpis chyb, které se v programu vyskytují.

Naneštěstí, některá chybová hlášení nejsou uvedena v manuálu. Po několika zkušenostech však brzy usoudíte, oč se jedná. Opravdu hezkou vlastností Kompilátoru je ta skutečnost, že až do konce programu pokračuje hledání chyb a konečný výsledek programu je vytištěn, a to dokonce i tehdy, když se skutečný proces kompilace zastaví. To vám hodně pomůže při odstraňování všech chyb předtím, než zkusíte program nově zkompilovat. Výpis chyb, právě tak jako výpis kódu assembleru, je možné zavést na obrazovku, případně nechat vytisknout na tiskárně, nebo obojí.

Kompilátor KYAN PASCALu má zabudovaný assembler a funkce spojování/řetězení, což je velmi výhodné. Jestliže jste vždy odsuzovali kompilátor za to, že jenom "vyplivuje" ven strojový kód určený ke spuštění programu a nic jiného, zde je přidán krok překladu assembleru (a spojování). Mnoho z vás jistě tuto vlastnost jen přivítá.

Sumárně vzato, cením si moc této soupravy. Dokonce cítím, že ze mě vyroste odborník na programování v jazyku PASCAL. Jazyk KYAN PASCAL není polovičatá verze PASCALu. Je to seriózní, plnohodnotná implementace jazyka PASCAL na 8-bitové počítače Atari XL/XE. Doporučuji ho všem bez výjimek. Nebudete zklamáni.

Jiří Hrdlička
Atari 130 XE

Action!
=======
OSS - Optimized Systems Software, Inc.
1221 B Kentwood Avenue
San Jose, California 95129
USA

Nový programovací jazyk pro 8-bitové počítače Atari XL/XE.
Neuvěřitelně rychlý vyšší programovací jazyk, který byl vytvořen pouze pro 8-bitové počítače Atari a který dokáže dokonale využít všechny jejich přednosti.
Action! je 200x rychlejší než Atari BASIC
11x rychlejší než Kyan Pascal
2.5x rychlejší než Turbo BASIC XL
1.5x rychlejší než BASIC XE.
Pokud tedy chcete pracovat s výkonným a strukturovaným typem vyššího jazyka, neváhejte a začněte programovat v jazyku Action!. Mimo jiné je to výborný úvod do programování v jazycích C nebo Pascal, a to i v případě přechodu na 16-bitový počítač Atari ST.

Action!
software

Action!

Předkládáme Vám upravený překlad článku z polského časopisu Bajtek, který vyšel roku 1988 jako jeho zvláštní příloha. Nemáme k dispozici žádnou firemní literaturu a svůj příspěvek nabízíme těm ataristům - programátorům, kteří rádi experimentují. Tento článek není a ani nemůže být žádnou učebnicí jazyka Action!, je myšlen jen jako stručná informace o této novince.

Action! je programovacím jazykem někde mezi Pascalem a jazykem C. Obsahuje řadu předností, které jej staví výše, než je BASIC. Programy napsané v Action! se svou rychlostí blíží k rychlosti práce počítače ve strojovém kódu a díky strukturování, editoru a bohaté knihovně programovacích procedur je programování poměrně snadné. Kompilátor Action! je dostupný na všech médiích systému Atari: na msf. kazetě (standart i TURBO), disketě i zásuvném modulu - cartridgi.

Systém Action! se skládá z editoru a kompilátoru, který kontroluje práci monitoru. Program píšeme pomocí editoru a

potom jej kompilujeme. Jeho spuštění je možné hned po zkompilování, a to i ve verzi zdrojové: monitor natáhne zdrojový program, zkompiluje jej a potom provede. Jistou slabinou kazetového systému Action! je nutnost jeho natažení před spuštěním zkompilovaného programu. To vyplývá z toho, že kompilátor nepřipojuje k výslednému programu knihovnu procedur. Ta je však v současné době již k dispozici.

Editor

Po spuštění programu se objeví v dolní části obrazovky jasnější řádek s nápisem Action! (c) 1983 ACS. Je to komunikační řádek, kde se objevuje hlášení chyb, jiné informace i doprovodné komentáře editoru. Při používání dvou okének editoru (viz dále) je komunikační řádek umístěn mezi nimi.

Editor Action! je systém umožňující snadné psaní programů. Dovoluje napsat řádek o délce až 240 znaků, přestože v okénku editoru je místo jen pro 38 znaků. Díky tomu je možné psát srozumitelně programy. Dosažení maximální délky řádku je signalizováno bzučákem. Když je řádek delší, než je šířka okénka, pak je krajní znak řádku zobrazen inverzně.

Text programu se do editoru píše bez nějakých speciálních symbolů. Abychom mohli psát kontrolní znak, stiskneme nejprve klávesu ESC. Pro editor to znamená, že následující znak má být interpretován jako text a ne jako příkaz editoru.

Způsoby práce editoru

Editor může pracovat dvěma způsoby - vsunováním nebo nahrazováním. Ve způsobu nahrazování nastupuje nový text místo stávajícího, ve způsobu vsunování je nový text dán mezi text stávající, a to bez jeho rušení. Po spuštění editor pracuje v režimu nahrazování, změna režimu je po příkazu SHIFT+CTRL+I.

Obsluha editoru

K odstranění textu z editoru se použije příkaz SHIFT+CLEAR. Ruší se tím nejen text na obrazovce, ale i celý obsah editoru. Při používání dvou komunikačních okének je však zrušen jen text, který patří k okénku, ve kterém se nachází kurzor. K zabránění neúmyslného zničení redigovaného programu se editor zeptá: "Clear?". Je třeba odpovědět buď Y (yes=ano) nebo N (no=ne). Když byly v redigovaném textu vykonány změny a nebyl zapsán, editor se opět zeptá: "Not saved - Delete?" (Není zapsáno - zrušit?), aby se ujistil, že opravdu nechceme upravenou verzi uložit k dalšímu zpracování. Existuje ještě jeden způsob opuštění editoru, a to kombinace SHIFT+CTRL+M. Tento příkaz zavádí kontrolu systému monitorem, odkud je možno volat jiné elementy systému nebo přejít do DOSu.

Program napsaný v editoru může být zaznamenán na vnějším zařízení. Umožní to příkaz SHIFT+CTRL+W. Pak se objeví nápis "Write?". Při zapisování na disk je nutné zadat název souboru, při zápisu na magnetofon stačí název periférie - "C:". Totéž platí i pro výpis programu tiskárnou - "P:".

Analogicky při nahrávání programu do počítače použijeme příkaz SHIFT+CTRL+R. Počítač se táže "Read?" a další postup je obdobný jako při ukládání programu.

Okénka

Jednou z možností, které nám nabízí editor Action!, je

vytvoření druhého komunikačního okénka a tedy současné redigované dvou programů. Druhé okénko se vytvoří příkazem SHIFT+CTRL+2. Tento příkaz rovněž slouží k přesunu kurzoru do druhého okénka. Zpět do prvého okénka se dostaneme příkazem SHIFT+CTRL+1. Pokud chceme zrušit okénko, umístíme do něj kurzor a stisknem kombinaci SHIFT+CTRL+D. Počítač napíše "Delete window?" (Smazat okénko?) a dále postupujeme jako při rušení obsahu editoru.

Okénko je možno posunovat nahoru a dolů pomocí pohybu kurzoru. Další příkazy navíc umožní přesun okénka vcelku nahoru nebo dolů. Při stisknutí SHIFT+CTRL+šipka nahoru se okénko umístí tak, že dřívější nejvyšší řádek je nyní nejnižším. Příkaz SHIFT+CTRL+šipka dolů posunuje okénko naopak. Použití příkazů SHIFT+CTRL+hranatá závorka pravá (respektivě levá) posune znak o jeden znak vpravo (respektivě vlevo).

Pohyb kurzorem

Níže uvedené příkazy vám umožní pohybovat kurzorem po celém prostoru editoru Action!:

CTRL+editační klávesy ... pohybuje kurzorem běžnými způsoby
SHIFT+CTRL+< ... posune kurzor na začátek aktuálního řádku
SHIFT+CTRL+> ... posune kurzor na konec aktuálního řádku
SHIFT+CTRL+H ... přemístí kurzor na začátek programu
SHIFT+TAB ... nastaví pozici tabelování
CTRL+TAB ... ruší pozice tabelování

Občas je užitečné rozdělit programovací řádek na řádky dva. Stačí pouze nastavit kurzor na znak, který má být prvním znakem nového řádku, a stisknout SHIFT+CTRL+RETURN. Opačně při spojování dvou řádků v jeden nastavíme kurzor na první znak druhého řádku a stisknem SHIFT+CTRL+DELETE/BACK SPACE.

Redigování textu

Editor Action! obsahuje několik funkcí, které bývají v běžných textových editorech. Jednou z nich je i výměna určitých fragmentů textu za jiné. Umožní to příkaz SHIFT+CTRL+S. Po jeho použití se na obrazovce objeví "Subtitute?" a pokud byla funkce už používána, vypíše se ostatní zaváděný "nový" text. Je třeba text napsat a stisknout RETURN. Samotné stisknutí klávesy RETURN (bez napsání nového textu) zachovává předchozí text. Nyní se na obrazovce objeví "for" (za), po kterém musíme napsat text, který má být nahrazen. Pokud již byla funkce použita, postupujeme analogicky - stejně jako s novým textem. Po stisknutí klávesy RETURN editor najde první výskyt "starého" textu a vymění jej za nový. Příkaz SHIFT+CTRL+S dovoluje změnu textu i bez dotazů "Substitute?" a "for", pokud byla předtím použita jiná funkce editoru. Pokud by starý text nebyl nalezen, vypíše se hlášení "not found" (nenalezeno).

Je možné i měnit nebo kopírovat celé bloky přes kopírovací zásobník - buffer. Po každém příkazu SHIFT+DELETE se ruší jeden řádek umístěný v zásobníku. SHIFT+CTRL+P přesune tento řádek na aktuální pozici kurzoru. Protože vsunutí textu nemaže buffer, je možné opravovat takto několikrát text.

Jestliže při redigování řádku uděláme chybu, můžeme si obnovit původní řádek příkazem SHIFT+CTRL+U pod podmínkou, že kurzor neopustil opravovaný řádek.

Značky (etikety)

Značky nám dovolují označit si libovolné místo v textu. Pokud chceme značku dát na určitou pozici kurzoru, pak použijeme příkaz SHIFT+CTRL+T. Napíše se dotaz "tag id:" - napíšeme jeden znak identifikační značky a stiskneme RETURN. Jestliže jsme tuto značku použili, pak je na starém místě zrušena a ponechána pouze na místě novém.

Příkaz SHIFT+CTRL+G nám přesunuje kurzor na určenou značku. Tu napíšeme po dotazu "tag id:". Jestliže uvedeme značku, která nebyla použita, na obrazovce se objeví hlášení "tag non set" (značka nebyla ustavena). Pozor - každá operace, která změní obsah řádku, zruší i jeho případnou značku.

Monitor

Monitor jazyka Action! řídí práci celého systému. Řádek příkazů monitoru se nalézá v horní části obrazovky a obsahuje znak ">" coby kurzor. Zbývající část obrazovky je prostor komunikační - má více významů. V čase běhu programu slouží k vysvětlování jeho výsledků, může též být využit ke sledování vykonávání programu. Pokud najde počítač v programu chybu, komunikační řádek vypíše číslo chybového hlášení a část s chybným řádkem.

Příkazy monitoru

Monitor identifikuje zadané příkazy pouze podle počátečního písmene. Proto je také zbytečné vypisovat celé řídící slovo. Vykonání příkazu následuje okamžitě po stisku klávesy RETURN.

B ... BOOT
- způsobí restart celého systému a návrat do editoru, ruší jak programy v paměti, tak proměnné

C ... COMPILE
- tento příkaz aktivuje kompilátor, který zkompiluje program uložený v editoru. Je možné také zkompilovat program, který je uložen na vnějším zařízení. Ovšem je nutné správně zadat název zařízení, např. u magnetofonu C"C:", u disketové stanice C"D:název.ext".

D ... DOS
- opuštění jazyka Action!, přechod do DOSu. Obsah editoru i monitoru je poškozen.

E ... EDITOR
- přechod do editoru jazyka Action!

O ... OPTION
- vyvolává tzv. menu volby, které dovoluje zdatnému programátorovi změnu parametrů práce editoru, kompilátoru i monitoru. Každá volba je vysvětlena v příkazovém řádku. Jestliže ji chceme změnit, napíšeme nový příkaz a stiskneme klávesu RETURN. Menu volby opustíme klávesou ESC. Podrobnosti viz další odstavec.

P ... PROCED
- obnoví vykonávání programu zastaveného stisknutím klávesy BREAK. Program je kontinuován, jako by nebyl přerušen.

R ... RUN
- spuštění zkompilovaného programu. Vystupuje ve čtyřech formách:

1) R - spuštění zkompilovaného programu z paměti počítače
2) R"C:" nebo R"D:název.ext" - natažení a spuštění programu z vnějšího zařízení
3) R"adresa" - spuštění od dané adresy
4) R"název procedury" - spuštění jedné procedury z programu nebo knihovny procedur
S ... SET
- příkaz umožní bezprostřední vstup do paměti RAM a eventuelně změnu jejího obsahu
W ... WRITE
- slouží k provedení zkompilovaného programu na periferní zařízení jako binární soubor. Program takto zapsaný na disk může být spuštěn DOSem (po natažení kompilátoru). Při zápisu na msf. kazetu je třeba uvést W"C:", na disketovou jednotku W"D:název.ext", na tiskárnu W"P:".
X ... XECUTE
- příkaz umožní vykonat libovolnou instrukci jazyka Action! nebo libovolné direktivy kompilátoru z monitoru (mimo příkazy MODULE a SET). Po příkazu je pak nutné zapsat instrukci k vykonání, např. XECUTE Print E("Welcome to Action!") (Vítej v jazyku Action!).
? ... znak otazník
- vysvětlí aktuální obsah paměti na dané adrese. V komunikačním okénku se napíše adresa v decimální nebo hexadecimální soustavě, znak ATASCII s kódem, který odpovídá obsahu paměťového místa a obsah dvou paměťových míst v hexadecimální soustavě nebo desetinné volby ve formátu klíčových slov BYTE a CARD.
* ... znak hvězdička
- vysvětlení aktuálního obsahu paměti od dané adresy. Vysvětlování lze zastavit stisknutím kombinace CTRL+1, opětným stiskem této kombinace se obnoví výpis.

Menu volby O (OPTION)

Všechny dostupné volby mají tento formát výpisu:
název volby - vysvětlení
Display?
- obrazovku je možné vypnout s cílem zvýšení rychlosti kompilace i operací I/O (vstupu/výstupu). Příkaz N obrazovku vypne, Y ji nechá zapnutou.
Bell?
- pokud se počítač potká s chybou při kompilaci, zazní tón bzučáku. Opět je možné vypnout příkazem N, příkaz Y tuto akustickou kontrolu znovu zapíná.
Case sensitive
- rozlišování písmen. Jestliže zvolíme Y, rozlišují se malá a velká písmena (např. SUMA, Suma a suma se liší). Instrukce jazyka Action! se musí psát vždy velkými písmeny. Po spuštění programu bývá tato volba vypnuta.
Trace?
- sledování. Díky tomu je možné sledovat kompilaci programu - volba Y. V komunikačním okénku jsou pak vysvětlovány všechny procedury spolu s parametry.
List?
- listing neboli výpis programu. Použitím příkazu Y se bude objevovat obsah právě kompilovaného řádku.

Window 1 size
- rozměr prvého okénka. Dáno na začátku programu. Rozměr druhého okénka je totiž závislý na velikosti prvého okénka, a to tak, že obě dohromady mají 23 řádků. Každé okénko tedy musí mít rozměr mezi 5-18 řádky.
Line size?
- rozměr řádku. Je to vlastně počet znaků na programový řádek, přičemž 240 znaků je maximum. Tato volba je velmi vhodná pro kontrolu tisku znaků na tiskárně.
Left margin?
- levý okraj. Levá hrana je oblast, od které se píší znaky na obrazovce. Může nabývat hodnot od 0 do 39.
EOL charactér?
- znak konce řádku (space-E). Je napsán prostřednictvím editoru jazyka Action! na konec každého řádku. Normálně je neviditelný (mezera), ale lze jej zviditelnit při redigování programu. Můžeme použít libovolný znak - nejlepší je však grafický, např. CTRL+T.

Spuštění programu

Po spuštění programu nás často opouští trpělivost při hledání "záhadných" chyb. Autoři jazyku Action! však mysleli i na tuto možnost a díky jim a možnostem monitoru Action! je to snadná činnost.
1) volba TRACE
Zkompilovaný a vykonávaný program je sledován na obrazovce. Název každé volané procedury se vypíše na obrazovku současně s odkazem na příslušné parametry.
2) zastavení kompilace
Spuštěný program je samozřejmě možné i zastavit. Jazyk Action! používá tyto dva způsoby:
a) klávesa BREAK
- funguje jen ve dvou případech, a to při operaci I/O a při vyvolání procedur s více jak třemi parametry
b) procedura BREAK
- zde je cílem kontrola chodu programu. Na několika vhodných místech v programu umístíme vyvolání procedury Break, která funguje podobně jako příkaz STOP v jazyku BASIC. Opětné uvedení programu do chodu provedeme příkazem P (Proceed).

Klíčová slova

Znalost každého jazyka je podmíněna dobrým seznámením s jeho klíčovými slovy; pak je programování snadnější. Klíčová slova mohou být samozřejmě používána jen správným způsobem a nelze jich používat jako názvů proměnných.

Bohužel, v časopisu Bajtek byl uveřejněn pouze výpis klíčových slov a symbolů bez bližšího vysvětlení:
AND - ARRAY - BYTE - CARD - CHAR - DEFINE - DO - ELSE - ELSEIF - EXIT - FI - FOR - FUNC - IF - INCLUDE - INT - LSH - MOD - MODULE - OD - OR - POINTER - PROC - RETURN - RSH - SET - STEP - THEN - TO - TYPE - UNTIL - WHILE - XOR
Používané symboly:

Symboly

Při popisu jazyka se používají různé symboly pro snazší pochopení jeho stavby.
Adresa - číslo paměťového místa v paměti počítače. Když počítači přikážeme, aby něco umístil do paměti, musíme mu udat adresu.
Identifikátor - název popisující proměnnou, proceduru atd. Název v jazyku Action! musí splnit následující podmínky:
- musí začínat písmenem
- ostatní znaky musí být buď písmena, číslice nebo znak podtržení ( )
- nesmí to být klíčové slovo.

Písmenem rozumíme jakékoliv písmeno malé nebo velké abecedy, číslice mohou být 0 - 9.
$ - znak dolaru. Je-li před číslem, pak toto číslo je v hexadecimální soustavě.
; - středník. Je symbolem pro komentář (ekvivalent REM v BASICu). Vše na řádku za ním kompilátor ignoruje.
< ... > - obsah v těchto ostrých závorkách znázorňuje, co musí být na tomto místě v instrukci nebo řádku programu. Např. <identifikátor> znamená, že zde musí být použit identifikátor. Pokud to může, ale nemusí být použito, označuje se takto - např. {<identifikátor>}. Podobné označení slouží ke znamení opakování, např. {:<identifikátor>:} znamená opakování libovolného počtu identifikátorů. Použití <identifikátor>*<adresa> značí: použij identifikátor nebo adresu, ale ne obě společně.

Typy dat

Před popisem typů dat, které používá jazyk Action!, si musíme povědět něco o konstantách a proměnných, protože to jsou základní objekty, se kterými počítač operuje.

Proměnné

Zvolený název proměnné musí být identifikovatelný podle pravidel. Pro název není žádné omezení. Doporučuje se (pro získání přehledného programu) označovat název proměnné raději delším názvem např. "celek" než pouhým písmenem "c". Na délku zkompilovaného programu nemá toto praktický vliv, jen se o něco málo prodlouží zdrojový program.

Konstanty

Action! používá tři typy konstant: číselné, znakové a konstanty kompilátoru.

Konstanty číselné jsou možné ve třech formách:
- decimální čísla - nevyžadují zvláštní popis
- hexadecimální čísla - mají před sebou typický znak - "$", např. $3000
- znaková čísla - značí se apostrofem (" ") před znakem, např. A

Znaky s číselnými konstantami jsou reprezentovány přes jednobitová čísla a jsou rovna kódu ATASCII znaku.

Konstanty kompilátoru dělíme na tři typy. Jsou používána během kompilování programu a na rozdíl od pevných proměnných, konstant, procedur nebo funkcí se při sestavování programu

nepoužívají. Konstanty kompilátoru mají následující formáty:
- číselný
- identifikátor
- ukazovatel
- kombinace některých dvou výše uvedených

Typy proměnných

V jazyku Action! vystupují tři typy proměnných: BYTE, CARD a INT. Všechny tři jsou číselné.

BYTE
- jde o čísla do 256, která jsou reprezentována jednobytovým číslem bez znaku, a to od 0 do 255. Díky tomuto může být BYTE používán jako znaková proměnná a umožňuje záměnu klíčových slov BYTE a CHAR.

CARD
- podobně jako BYTE, ale s větším rozsahem. Vnější reprezentace je dvoubytová, čísla tedy jsou od 0 do 65535. Čísla typu CARD mohou být uložena ve formátech LSB, MSB.

INT
- celá čísla se znaménkem, a to od -32768 do 32767.

Deklarace

Před každým použitím identifikátoru je nutné jej deklarovat. Počítač si pro něj musí rezervovat určité místo v paměti. Formát deklarace je: typ - identif.-inform.poč.-identif.-inform.poč. Deklaraci proměnných je třeba uvést v programu bezprostředně po instrukci MODULE nebo na počátku procedury nebo funkce. Např.:

BYTE start, end
- deklarace proměnných start, end jako typ BYTE

INT num=0
- proměnná typu INT velikosti 0

BYTE x=$8000
- proměnná x typu BYTE umístěná v paměti na adrese $8000

Programovací jazyk Action! a program Action!Library (knihovna) je k dispozici v Atari klubu v Bruntále. Programy v tomto jazyku začíná také otiskovat polský měsíčník Bajtek.

MUDr.Alois Rozsípal
AK Bruntál

Příkazy editoru Action!

Přechod do monitoru	SHIFT+CTRL+M (monitor)
Příkazy I/O:	
Čtení souboru	SHIFT+CTRL+R (read)
Čtení directory disku	SHIFT+CTRL+R ?n:*.*
Zápis souboru	SHIFT+CTRL+W (write)
Výpis programu na tiskárnu	SHIFT+CTRL+W P:
Pohyby kurzorem:	
Kurzor nahoru, dolů, vpravo, vlevo	CTRL+
Kurzor na začátek řádku	CTRL+<
Kurzor na konec řádku	CTRL+>
Ukončení řádku	RETURN
Tabulace	TAB
Ustavení tabulace	SHIFT+TAB
Zrušení tabulace	CTRL+TAB
Pohyb okénka:	
Počátek souboru	SHIFT+CTRL+H (head)
První okénko nahoru	SHIFT+CTRL+
První okénko dolů	SHIFT+CTRL+
O jeden znak vlevo	SHIFT+CTRL+>
O jeden znak vpravo	SHIFT+CTRL+<
Vytvoření druhého okénka	SHIFT+CTRL+2
Přechod do prvního okénka	SHIFT+CTRL+1
Přechod do druhého okénka	SHIFT+CTRL+2
Zrušení okénka	SHIFT+CTRL+D (delete)
Redigování textu programu:	
Změna režimu vsunutí/nahrazení	SHIFT+CTRL+I (insert)
Vymazání změněného řádku	SHIFT+CTRL+U (undone)
Přisunutí původního řádku	SHIFT+CTRL+P (paste)
Zrušení řádku v bufferu	SHIFT+DELETE
Vsunutí textu z bufferu	SHIFT+CTRL+P (paste)
Vyhledávání	SHIFT+CTRL+F (find)
Výměna	SHIFT+CTRL+S (substitute)
Rozdělení řádku na dva	SHIFT+CTRL+RETURN
Spojení dvou řádků v jeden	SHIFT+CTRL+DELETE
Definování značky (etikety)	SHIFT+CTRL+T (tag set)
Vyhledávání značky (etikety)	SHIFT+CTRL+G (go to tag)

Příkazy monitoru Action!

B ... boot	restart Action!
D ... DOS	přechod do DOSu
E ... editor	přechod do editoru
O ... option	výběr možných variant
X + instrukce ... xecute	vykonání instrukce
C ("název") ... compile	kompilace programu
W ("název") ... write	zápis programu
R ("název") ... run	spuštění programu
P ... proced	kontinuování programu
SET ("adresa") (obsah)	ustavení obsahu adresy paměti (=POKE)
? ("adresa")	obsah adresy paměti (=PEEK)
* ("adresa")	přehled paměti

Chybové kódy Action!

Číslo chyby:		Význam:
0	...	nedostatečná kapacita paměti
1	...	chyba v řetězci
2	...	chybná direktiva DEFINE
3	...	chyba v tabulce symbolů - globální proměnné
4	...	chyba v tabulce symbolů - místní proměnné
5	...	chyba v direktivě SET
6	...	nesprávná deklarace
7	...	příliš mnoho argumentů v proceduře nebo instrukci
8	...	není deklarována proměnná
9	...	užito proměnné místo konstanty
10	...	zakázaná instrukce
11	...	syntaktická chyba
12	...	chyba v instrukci THEN
13	...	chyba v instrukci FI
14	...	malý prostor paměti pro výsledný kód
15	...	chyba v instrukci DO
16	...	chyba v instrukci TO
17	...	nesprávný formát výrazu
18	...	neuzavřená závorka
19	...	chyba v instrukci OD
20	...	není možný přesun v paměti
21	...	nedovolená tabulka
22	...	velký vstupní soubor
23	...	nedovolená podmínka
24	...	chyba v instrukci FOR
25	...	nedovolená instrukce EXIT
26	...	použito víc než 16 hladin
27	...	chyba v instrukci TYPE
28	...	zákaz příkazu RETURN
61	...	chybné místo v symbolické tabulce
128	...	zastavení programu klávesou BREAK

Dvojí podoba počítače ATARI 130 XE

Banky paměti RAM na počítači Atari 130 XE
Přístup k plnému využití 128 kB RAM

Pro větší srozumitelnost textu jsem nakreslil diagram znázorňující organizaci paměti počítače Atari 130 XE, který by vám měl pomoci všemu porozumět. Prosím, prostudujte si ho a teprve potom pokračujte dále. Pomocí diagramu si tedy představte Atari 130 XE jako 64 kB počítač s několika přidanými "extra" bankami paměti RAM.

Proč 64 kB počítač? Zcela jednoduše - máte k dispozici právě tolik paměti, kolik může adresovat 8-bitový mikroprocesor 6502. Technické zdůvodnění: mikroprocesor používá 16 adresových řádků a 2 na 16 je 65536 (počet možných jedinečných adres) - což je 64 kB.

Jaká je tedy vnitřní sestava tohoto 64 kB počítače? Prostudujte si levý sloupec na diagramu. Zjistíte, že sestava obsahuje 10 kB operačního systému Atari (OS), 2 kB oblasti I/O (vstup/výstup) atd. Jestliže sečtete všechno dohromady, dostanete právě 64 kB. Při zapnutí počítače obdržíte vždy tuto konfiguraci (pokud ovšem nepodnikáte nic speciálního). To znamená, že Atari BASIC je "zapnut" a je plně k vaší dispozici.

Existují tři cesty, jak pozměnit tento stav.

Za prvé, do počítače můžete vložit cartridge (viz příslušný konektor vzadu). Na vnitřní struktuře cartridge závisí, jestli je 8 kB (např. Editor/Assembler cartridge) nebo 16 kB (např. Atari Writer cartridge) paměti RAM věnováno cartridgi. Upozorňuji, že počítač Atari 130 XE nemůže bez úprav hardwaru potlačit nebo vyřadit cartridge z činnosti. (Pouze tzv. supercartridge firmy OSS mají speciální obvody, které připouští vypnutí vlastní cartridge přes software. Ale tyto obvody jsou pouze v cartridgi, nikoli v počítači Atari 130 XE.)

Za druhé, podobným způsobem pracují hardwarová zařízení, která se připojují k počítači přes konektor cartridge a konektor ECI (Enhanced Cartridge Interface). Tato zařízení mohou "vyprázdnit" oblast normálně používanou OS Atari (např. matematickými operacemi s plovoucí čárkou). Takto pracuje např. hardisk firmy SupraDrive a několik zařízení firmy ICD - vždy mění mapu paměti počítače Atari 130 XE.

Konečně, nejlehčí cesta k řízení volby banky "extra" RAM paměti (najděte si tato označení na diagramu) je přes hardwarový řídící port na adrese $D301 - PORTB. Nebudu se teď zabývat bity, které řídí tento port. Příslušnou část najdete v příloze k tomuto článku (překlad z angl. - Atari 130 XE Owner s Manual, str. 121-122). Pokračujme tedy dál...

Je to jednoduché!?

Naznačil jsem, že "nejlehčí cesta" vede přes řídící port PORTB. Jedině tak máme možnost použít více paměti RAM, než nám dává standartní konfigurace 64 kB počítače. Ale ačkoliv je to nejjednodušší, není to vůbec lehké! Toto tvrzení není v

rozporu: 90 % stupeň obtížnosti je přece jen snadněji řešitelný než 95 nebo 98 % stupeň, i když je to stále pracné.

Moje doporučení: pokud neznáte assembler 6502 nebo se ho učíte, nezaplétejte se zatím s bankami RAM. Je totiž velmi jednoduché "zmrazit" činnost počítače a tím nenahraditelně ztratit programy. Také se vám může stát, že omylem vypnete tu část paměti, kterou zabírá váš program. V tomto případě CPU okamžitě způsobí kompletní ztrátu celého programu. Také se vám může podařit vypnout OS - nastane přerušení a počítač se "uloží ke spánku".

Banky extra paměti RAM jsou rozmístěny mezi adresami $4000 a $8000. Dbejte, aby se všechny části vašeho programu vyhnuly této oblasti.

Pokud trochu ovládáte programování ve strojovém jazyku nebo assembleru (nebo snad jazyku ACTIO!), můžete vyzkoušet a využít výhody bank extra paměti RAM, které se nacházejí mezi adresami $4000 a $8000. Počínejte si velmi opatrně, protože program, který přepíná banky extra RAM, nesmí zasahovat do žádné z těchto oblastí. Také je zakázáno používat data uložená v jedné bance RAM, zatímco druhá paměťová banka je stále aktivní.

Ale stále se vyhýbejte oblasti paměti RAM od adresy $C000 výše. Protože tato oblast leží stranou od různých obtížných manipulací jako je přerušení, znakové sady atd., stále více a více verzí diskového operačního systému využívá této oblasti paměti. Tím šetří místo pro programátora ve snadněji přístupné oblasti "nižší paměti". DOS XL nebo SpartaDOS již zabírají tuto část paměti RAM a také nový diskový operační systém A-DOS pro disketovou jednotku XF-551 plně využívá tuto možnost.

Řešení v BASICu

A co mají dělat programátoři v BASICu? Jak oni mohou použít banky paměti "extra" RAM na počítačích Atari 130 XE? Existují dvě dobré cesty:

1) Používejte komerční (standartní) program RAMdisk. Třeba ten, který je přiložen k soupravě DOS 2.5. Budete spokojeni...

2) Zkuste sehnat jazyk BASIC XE od firmy OSS. Je to jediný jazyk vytvořený pro specifické použití na počítačích Atari 130 XE. Dovoluje využít tolik paměti RAM, kolik je jen možné. Můžete psát programy o délce až 62 kB, a to vám ještě zůstane 32 kB na různé data-proměnné, které obsahují řetězce a pole. Máte k dispozici nejen všechny tyto možnosti, ale i 98 % kompatibilitu s jazykem Atari BASIC. To je pro velice výhodné, protože se už nemusíte učit všechno od začátku, ale navazujete na vědomosti a zkušenosti s jazykem Atari BASIC. Jen si příslušně rozšíříte možnosti - BASIC XE se totiž velice podobá jazyku Turbo BASIC XL. (Malá poznámka na závěr: víte, že BASIC XE je 2 až 6 krát rychlejší než Atari BASIC? Dokonce rychlejší než IBM PC?)

Než se pustíte do práce...

Shrňme to: špatné zprávy jsou ty, že napsání programu, který by využíval "extra" paměť RAM na počítači Atari 130 XE, není vůbec jednoduché (snad s výjimkou jazyka BASIC XE). Dobré zprávy jsou pak ty, které sdělují, že máte vždy přístup k

dalším 64 kB paměti RAM. Jen se to naučit využívat. K tomu vám snad pomůže následující text.

Jak střídat osm obrázků rychlostí blesku?

130 XE Slide Show

Naučte se použít extra 64 kB paměť na svém počítači Atari 130 XE. Tyto dva krátké programy vám budou demonstrovat použití jazyka Atari BASIC ke spínání jednotlivých bank paměti RAM. První program, Bank Switcher, vyžaduje pouze počítač Atari 130 XE nebo jiný model firmy Atari s kompatibilním rozšířením paměti. Druhý program, Slide Show, je náročnější - bezpodmínečně je nutná disketová jednotka (nejlépe Atari 1050), zatímco u programu Bank Switcher postačí kazetový magnetofon (např. XC-12).

Jak mohu použít rozšířenou paměť RAM ve svých BASICových programech?

To je jedna z otázek, na kterou se nejčastěji ptají majitelé počítače Atari 130 XE. V předchozím článku jsme si už vysvětlili obtíže spojené s použitím Atari BASICu k řízení spínání a přepínání bank paměti RAM. Doufám, že jste si pozorně celý předchozí text pročetli. Pokud ne, odskočte si na začátek a teprve potom se vraťte sem - k vlastní technice použitelné v programech.

Ačkoliv programy pro přepínání a spínání bank je nejlepší psát v jazyku assembler, pro naše účely demonstrace techniky použití snadno postačí i jazyk Atari BASIC, který je dostatečně výkonný. Následující dva demonstrační programy jsou sice krátké, ale neobyčejně efektivní.

Tyto programy mají i praktické použití, zvláště program Slide Show. Ten je ideální zvláště pro různé akce, setkání, demonstrační ukázky nebo krátké animované sekvence.

Bank Switcher

Natypujte do počítače listing 1 - program SWITCH.BAS a zkontrolujte ho pomocí programu TYPO II (viz Atari zpravodaj Olomouc 3-4). Před spuštěním program uložte na magnetofonovou kazetu (CSAVE) nebo na disk (SAVE "D:SWITCH.BAS").

Tento krátký demonstrační program zaplní první "extra" banku paměti RAM písmenem "A", druhou banku paměti zaplní písmenem "B", třetí banku písmenem "C" a čtvrtou písmenem "D". Plní vždy celou banku - tj. 16 kB RAM. Program potom "přepíná" udivující rychlostí obsah paměťových bank 0-3. Obsah těchto bank se nám vždy objeví na obrazovce v grafickém modu 2.

Každá obrazovka je uložena v jiné 16 kB bance rozšířené paměti RAM počítače Atari 130 XE. Napište příkaz RUN a odešlete jej. Nejprve uvidíte, jak počítač zaplňuje postupně každou banku paměti RAM odpovídajícím písmenem. Potom se písmena budou měnit před vašimi zraky podle toho, která z bank rozšířené paměti RAM bude "zapnuta". Vše probíhá velice rychle - to je hlavní přínos velké kapacity paměti RAM počítače Atari 130 XE

Slide Show

První program byl spíše jen ukázkou možností počítače Atari 130 XE. Druhý program - Slide Show - je již schopen praktického použití. I když není o mnoho delší než program Bank Switcher, je o mnoho užitečnější.

Natypujte listing 2 - program SLIDES.BAS. Opět použijte program TYPO II ke správnému přepsání programu. Upozorňuji, že tento program vyžaduje disketovou jednotku. (Majitelé magnetofonů mají zase smůlu...) Před spuštěním programu příkazem RUN program uložte na disk (SAVE "D:SLIDES.BAS).

Program Slide Show dokáže uložit osm vašich nejhezčích obrázků do rozšířené paměti RAM. Vytvoří tak vlastně digitální diapozitivy. Tyto můžeme měnit resulovatelnou rychlostí, což je značná výhoda proti standartnímu nahrávání obrázků z disku, které je značně pomalé. Pokud si to totiž budete přát, obrázky se mohou střídat oslepující rychlostí.

Má to jen jednu nevýhodu - k vytvoření obrázků musíte použít buď programu Micro-Painter nebo programu PD Micro Paint Artist. Jestliže používáte jiný kreslící program, např. Koala MicroIllustrator, nezoufejte si. Existují konvertující programy, které převádí záznamové formáty obrázků mezi sebou - např. program PICCON20. Tento program se mi velice osvědčil, spolehlivě převádějí obrázky na formát programu Micro-Painter

Až si vyberete nebo nakreslíte obrázky, které chcete použít v programu, máte dvě možnosti:
1) Všech osm obrázků nakopírujte pod názvy "D:F1", "D:F2" až "D:F8" na disk, na kterém již máte soubor SLIDES.BAS, tedy program Slide Show.
2) Pokud chcete použít své názvy obrázkových souborů nebo svůj disk s obrázky, zeditujte řádky 180-210 v programu Slide Show. Názvy souborů zde nahraďte vlastními.

Teď už jen odstartujte program příkazem RUN. Program Slide Show uloží nejdříve osm obrázků z disku do rozšířené paměti RAM. Nemějte obavy, pokud uvidíte jen čtyři obrázky při nahrávání. Vzniklo to tím, že počítač ukládá vždy dva obrázky do jedné paměťové banky RAM. Až se všechny obrázky uloží do správných bank, provede se automatická korekce barev a "show" může začít.

Rychlost střídání jednotlivých obrázků volíme ovladačem paddle, který zapojíme do joystickového portu 1. Poměrně snadno lze program předělat na joystickové řízení rychlosti. Je to snadné i pro méně zkušeného programátora, proto tuto úpravu přenechávám fantazii a schopnostem uživatele.

Jak program pracuje?

Každý obrázek, který má formát programu Micro-Painter, obsahuje 7680 bytů informací o stavbě obrázku a 4 byty, které program informují o použitých barvách. Dohromady to dává 7684 bytů, což je méně než 8 kB paměti. Můžeme proto vzít vždy dva tyto obrázky a umístit je spolu do jedné 16 kB banky "extra" paměti RAM. Kombinací snímkového stránkování a techniky přepínání bank paměti tak dokážeme využít téměř většinu rozšířené paměti RAM počítače Atari 130 XE!

Na starších modelech 8-bitových počítačů Atari (Atari 400

nebo Atari 800 - počítače se 4 joystickovými porty) byly joystickové porty 3 a 4 řízeny návěštím PORTB (adresa 54017, $D301). U počítače Atari 130 XE, který má pouze dva joystickové porty, používáme PORTB k vymezení množství paměti RAM, které bude používáno procesory CPU a ANTIC.

V prvé řadě se zajímáme o bity 2 až 5 návěští PORTB. Bity 0, 6 a 7 nastavíme tak, aby daly základní hodnotu - číslo 193. Budou tedy rovny jedné, protože 1+64+128 je rovno číslu 193. Nula v bitu 5 upozorňuje procesor ANTIC, že chceme použít rozšířenou paměť RAM. Totéž platí pro bit 4, který takto upozorňuje procesor 6502 (CPU). Tyto bity 4 a 5 řídí v programu proměnná MODE. Co se týká bitů 2 a 3, ty určují, která ze čtyř bank rozšířené paměti RAM bude použita.

Když už toto známe, můžeme použít ve svých programech následující vzorec, který nám umožní spínat a přepínat jednotlivé banky paměti RAM:

POKE PORTB,(193+16*MODE+4*BANK)

V programu Slide Show je proměnná MODE nastavena na hodnotu 0, což, jak už víme, připravuje procesory ANTIC a CPU k použití rozšířené paměti RAM. Proměnné BANK0 až BANK3 obsahují správnou hodnotu nastavení banky RAM. Tyto proměnné jsou definovány na řádcích 278-280. Inicializační rutina display listu je umístěna na řádcích 250 a 300-350.

Jestliže tedy používáme mod rozšířené paměti RAM, musíme si uvědomit, že paměť RAM od adresy 16384 ($4000) po adresu 31767 ($7FFF) je "vypnuta" a na její místo byla přemístěna jedna ze čtyř 16 kB bank. Když tedy použijete příkaz PEEK nebo POKE, který by měl zasahovat do této oblasti paměti, upozorňuji vás, že bude ve spojení s novou bankou paměti RAM místo původní, která patří do standartní konfigurace 64 kB paměti RAM počítačů XL/XE. Dávejte na to pozor!

Program Slide Show ukládá vždy dva obrázky do každé ze čtyř 16 kB bank rozšířené paměti. Řádky 290 a 360-460 spínají příslušnou paměťovou banku a potom volají podprogram (řádek 740) k uložení dvou obrázků do této banky RAM.

Byty 4 a 5 display listu vytvářejí adresu, která nám ukáže začátek obrazové paměti. Podotýkám, že byte 4 (nižší adresa bytu) je nastaven příkazem POKE na hodnotu 16 (viz řádek 300). Toto vyvážení je potřebné, neboť naše obrazovka dlouhá 7680 bytů musí nutně překřížit 4 kB hranici v paměti RAM. Bez vyvážení by se spodní část obrázku neobjevila správně na TV obrazovce. Podprogram, který nahrává obrázkové soubory, vyžaduje totéž (viz řádek 770).

Další proměnná HI je použita jako vyšší adresa bytu obrazové paměti. Je uložena pomocí příkazu POKE jako příslušná hodnota do display listu. Během procedury nahrávání obrázků z disku však také proměnná HI rozhoduje, zda bude obrázek uložen do vyšších nebo nižších 8 kB RAM banky rozšířené paměti počítače Atari 130 XE.

Na řádcích 470-620 programu Slide Show najdeme spínání a přepínání bank spolu se snímkovým stránkováním. Používají poprogram na řádku 630 k nastavení barevných registrů a upravení display listu.

To je pro dnešek vše. V některém z dalších čísel Atari zpravodaje Olomouc uveřejním další příklady a možnosti využití

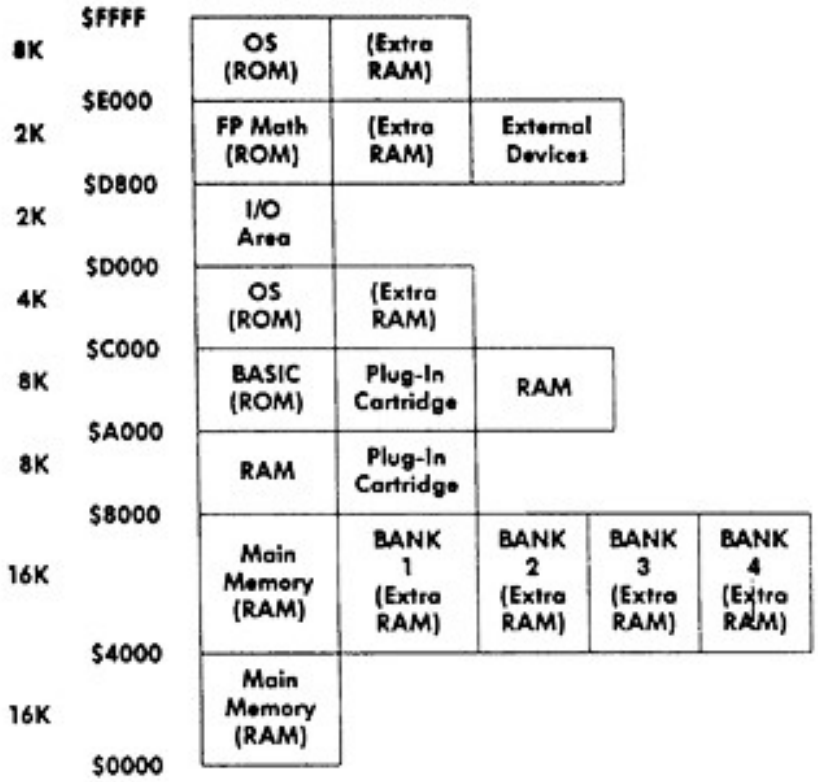
rozšířené paměti RAM na počítači Atari 130 XE. Pokud však nechcete čekat a máte možnost pořízení literatury ze zahraničí, doporučuji spolu s autory tyto tituly:
Lon POOLE : Your Atari Computer
(Osborne/McGraw-Hill)
Jeffrey STANTON, Dan PINAL : Atari Graphics and Arcade Game Design
(The Catalog)

(c) 1988 by GIA Software
Jiří Hrdlička
Atari 130 XE

Literatura:
1. Nat Friedland : 130 XE Double Feature
2. Bill Wilkinson : RAM Banks on the 130 XE
3. Bill Marquardt : 130 XE Slide Show
4. Atari 130 XE Owner s Manual

Časopisy Analog, Antic a COMPUTE! 1987/88

FIGURE 1
130XE MEMORY MAP



Příloha 1:

Využití plného potenciálu paměti RAM počítače Atari 130 XE

Počítač Atari 130 XE obsahuje 131 072 bytů paměti s libovolným výběrem (RAM) - to je dvakrát více než Atari 800 XL (800 XE nebo 65 XE). Ve většině případů je však polovina této paměti (65 536 bytů) pro uživatele pouze transparentní - tj. nedostupná.

Tuto paměť je možné využít k uložení rozsáhlejší soustavy dat buď pomocí programového vybavení (programy SynPower, Atari Writer+, PaperClip 2.0 atd.) nebo pomocí diskového operačního systému DOS 2.5 (pouze majitelé disketových jednotek Atari 1050 nebo XF 551). Pomocí DOS 2.5 simuluje přídavná paměť funkci velmi rychlé disketové jednotky, tzv. RAMdisku. (Další informace o Atari 130 XE a DOS 2.5 najdete v příručce "DOS 2.5 - Disketová jednotka Atari 1050".)

Pomocí programového vybavení můžete využít rozšířenou paměť RAM i v Atari BASICu. Docilujeme toho pomocí metody přepínání paměti. Jestliže použijeme tuto metodu, je počítač přinucen pracovat s větším rozsahem paměti, i když čipy centrálního procesoru 6502 a videoprocesoru ANTIC mohou přímo adresovat jen 65 536 bytů paměti RAM. Přídavná a základní paměť je rozdělena vždy na 4 bloky po 16 kB.

Programem řízený přepínač paměti může kdykoliv nahradit druhou banku základní paměti umístěnou od adresy 16384 po adresu 32767 (hexadecimálně $4000-$7FFF) libovolnou bankou přídavné paměti. Stav přepínače přitom určuje, kterou částí (bankou rozšířené paměti) bude tato oblast nahrazena. Přepínač bank je umístěný na adrese 54017. Tato paměťová buňka je použita jako adresa vstupního kanálu mikroprocesorem 6502. Má tedy funkci interfacu pro periférie, který řídí vstup a výstup počítače.

Bity 4 a 5 určují, který z procesorů počítače bude mít přístup k bance rozšířené paměti. Normálně jsou tyto bity nastaveny na hodnotu 1 - oba procesory používají základní konfiguraci paměti pro 8-bitové počítače. Bit 4 umožňuje přepnutí paměti pro CPU a bit 5 provádí totéž pro ANTIC. Nastavení bitu 4 na nulovou hodnotu umožní přístup k rozšířené paměti RAM pro CPU. Analogicky platí totéž pro bit 5 a ANTIC. Pokud tedy nastavíme oba dva bity na nulovou hodnotu, nastane změna a oba procesory mohou začít používat přídavnou paměť.

Bity 2 a 3 určují, která část paměti bude využívána. Tyto dva bity dovolují celkem čtyři různé kombinace konfigurace paměti, kterou představují čtyři různé 16 kB banky z paměťové oblasti rozšířené paměti RAM.

Obsah paměťové adresy 54017 je při normálním provozu 193. Pomocí jazyka Atari BASIC a hlavně příkazu POKE můžete obsah tohoto paměťového místa modifikovat a přepínat tak paměťové banky počítače Atari 130 XE.
Např. POKE 54017,255 vybírá první banku rozšířené paměti RAM a tato paměťová banka je přístupná pro CPU, ne však pro videočip ANTIC.

Vzorec pro určení správné banky přídavné paměti RAM :
(platí pouze pro počítač Atari 130 XE)

```
POKE 54017,193+4*ADDRESS+16*MODE
```

Tabulka hodnot :

proměnná ADDRESS	oblast paměti
0	0 - 16383
1	16384 - 32767
2	32768 - 49151
3	49152 - 65535

proměnná MODE	přístup 6502	přístup ANTIC
0	extra	extra
1	normal	extra
2	extra	normal
3	normal	normal

Paměťová adresa 54017 :

bit # :	7	6	5	4	3	2	1	0
hodnota :	128	64	32	16	8	4	2	1
nastavení :	1	1	A	B	C	D	0	1

A ... Video Bank / Enable (VBE)
B ... CPU Bank / Enable (CBE)
C ... Bank MSB
D ... Bank LSB

Bity 0, 6, 7, 1 by měly být vždy nastavené. Jestliže je nastavený bit 4 (CBE) nebo bit 5 (VBE), potom zodpovídající procesor má přístup k základní paměti. Pokud jsou tyto dva bity nastaveny na nulu - na adresách $4000-$7FFF se nachází ta banka rozšířené paměti RAM, která je určena bity 2 a 3.

Příloha 2:

Test rozšířené paměti RAM počítače Atari 130 XE

Existuje ještě jedna vážná otázka pro majitele počítačů Atari 130 XE. Jak zjistit správnou funkci každého bytu rozšířené paměti RAM? U počítačů Atari řady XL/XE se testuje většina nejdůležitějších funkcí tzv. selftestem počítače. Mezi testované funkce patří sice ověření správné činnosti paměti RAM, ale pouze paměti patřící do základní konfigurace 64 kB počítače (viz diagram). Ale počítač Atari 130 XE má dvojnásobek této paměti! Jak tedy zkontrolovat stav rozšířené paměti RAM?

Na tuto otázku nám dá odpověď testovací program pro tuto oblast paměti. Proto neváhejte a natypujte listing 3 - program RBTXE.BAS. Program zkontrolujte opět pomocí kontrolních kódů programu TYPOII. Před vlastním spuštěním program uložte na magnetofonovou pásku (CSAVE) nebo na disk (SAVE "D:RBTXE.BAS").

Program není třeba nijak popisovat. Po spuštění příkazem RUN totiž program pracuje sám a výsledek oznámí asi po 30 minutách na obrazovce. Počítač sice kontroluje každý byte paměti RAM, ale na obrazovku vypisuje test vždy po 1 kB - čtverec velikosti kursoru. I tak se dá snadno lokalizovat chyba určitého paměťového místa.

Upozornění zejména pro majitele disketových jednotek. Program při testu rozšířené paměti RAM současně ruší veškeré informace uložené v této oblasti paměti. Proto pozor na používání jakkýchkoliv typů RAMdisku, zejména na uložení programů a dat v RAMdisku, které při činnosti testu nenahraditelně ztrácíme.

Program vznikl na základě článku publikovaného v časopise COMPUTE!. Je upraven tak, aby si uživatel mohl snadno prohlédnout funkci a činnost programu již při letmém pohledu na jeho listing. Autoři článku doporučují každému majiteli počítače Atari 130 XE prověřit rozšířenou paměť RAM pomocí tohoto programového testu. Toto doporučení mohu jen podpořit. Ostatní záleží na vás...

```
EB 10 REM ----------------------
PH 20 REM I RAM Banks Tester I
EV 30 REM ----------------------
TO 40 REM - only for ATARI 130 XE
XE 50 REM - (c) 1988
BM 60 REM - by GIA Software
BF 70 REM Extended RAM Banks Memory
VP 80 REM It takes aprox. 30 min.
BG 90 REM
SG 100 GRAPHICS 0
FI 110 DL=PEEK(560)+PEEK(561)*256+4
YN 120 POKE DL-1,71:POKE DL+2,6
RN 130 POSITION 4,0:? "ATARI 130 XE"
TJ 140 POSITION 21,0:? "extended ram test
   er"
CL 150 POKE 710,0:POKE 752,1:POKE 82,1
UR 160 ? :? "Checking...          ( aprox
   . 30 min. )"
CB 170 ? CHR$(160);"  1 kB - O.K.
     ";CHR$(248);"  1 kB - Error"
CU 180 ?
EK 190 I=0:C=0:E=0
QT 200 FOR BANK=0 TO 3
RY 210 POKE 54017,BANK*4+193
TK 220 FOR X=16384 TO 32768
JE 230 POKE 77,0
ZC 240 POKE X,255
XU 250 IF PEEK(X)<>255 THEN I=1
NT 260 C=C+1
RZ 270 IF C=1024 THEN GOSUB 500
MC 280 NEXT X
IQ 290 ? CHR$(32);16384+(16384*BANK):?
AB 300 NEXT BANK
BL 310 ? :? :? "Result :":?
GD 320 IF E=0 THEN ? "Extended Memory is
   O.K.":END
LL 330 ? "Error in Extended Memory ( ";E;
   " loc. )":END
TK 500 IF I=1 THEN ? CHR$(248);CHR$(32);:
   E=E+1:GOTO 2020
QC 510 ? CHR$(160);CHR$(32);
MF 520 I=0:C=0
ZH 530 RETURN
```

Listing 2

```
QB 10 REM ------------------------------
RY 20 REM I 130 XE BANK SWITCHING DEMo I
QB 30 REM ------------------------------
TR 40 REM by Bill Marquardt
QP 50 REM (c) 1987
EC 60 REM ANTIC Publishing
EY 70 REM program SWITCH.BAS
OK 80 REM only for computer
VQ 90 REM Atari 130 XE
ZC 100 GRAPHICS 18
VC 110 GOSUB 380
AN 120 POKE PORTB,193+MODE+BANK0
VW 130 POKE DL+4,0:POKE DL+5,64
FF 140 FILLER=33:GOSUB 340
BW 150 POKE PORTB,193+MODE+BANK1
FW 160 FILLER=34:GOSUB 340
DD 170 POKE PORTB,193+MODE+BANK2
GN 180 FILLER=35:GOSUB 340
EK 190 POKE PORTB,193+MODE+BANK3
GL 200 FILLER=36:GOSUB 340
AM 210 POKE PORTB,193+MODE+BANK0
HY 220 FOR D=1 TO 100:NEXT D
BT 230 POKE PORTB,193+MODE+BANK1
IC 240 FOR D=1 TO 100:NEXT D
DA 250 POKE PORTB,193+MODE+BANK2
IG 260 FOR D=1 TO 100:NEXT D
EH 270 POKE PORTB,193+MODE+BANK3
IK 280 FOR D=1 TO 100:NEXT D
MU 290 GOTO 210
GJ 300 POKE PORTB,NORML
EA 310 POKE DL+5,INT(SC/256)
KW 320 POKE DL+4,SC-(PEEK(DL+5)*256)
NY 330 END
GL 340 FOR I=0 TO 239
CV 350 POKE NUSCREEN+I,FILLER
GF 360 NEXT I
ZN 370 RETURN
ND 380 PORTB=54017
OE 390 NORML=PEEK(PORTB)
UC 400 DL=PEEK(560)+PEEK(561)*256
CS 410 SC=PEEK(DL+4)+256*PEEK(DL+5)
QB 420 NUSCREEN=16384
YO 430 MODE=0
VH 440 BANK0=0:BANK1=4:BANK2=8:BANK3=12
ZK 450 RETURN
```

Listing 3

```
QB 10 REM ------------------------------
RY 20 REM I 130 XE BANK SWITCHING DEMo I
QB 30 REM ------------------------------
TR 40 REM by Bill Marquardt
QP 50 REM (c) 1987
EC 60 REM ANTIC Publishing
WM 70 REM program SLIDES.BAS
OK 80 REM only for computer
VQ 90 REM Atari 130 XE
SG 100 GRAPHICS 0
VI 110 DIM F1$(15),F2$(15)
XO 120 DIM F3$(15),F4$(15)
ZY 130 DIM F5$(15),F6$(15)
CA 140 DIM F7$(15),F8$(15)
GX 150 F1$="D:F1":F2$="D:F2"
LP 160 F3$="D:F3":F4$="D:F4"
QH 170 F5$="D:F5":F6$="D:F6"
UZ 180 F7$="D:F7":F8$="D:F8"
VL 190 DIM F$(15)
PH 200 GRAPHICS 15+16
MO 210 PORTB=54017
UE 220 DL=PEEK(560)+PEEK(561)*256
YM 230 MODE=0
ET 240 BANK0=0:BANK1=4
UM 250 BANK2=8:BANK3=12
AW 260 POKE PORTB,193+MODE+BANK0
ID 270 POKE DL+4,16:POKE DL+5,64
KM 280 FOR I=6 TO 196
VB 290 POKE DL+I,14:NEXT I
AH 300 POKE DL+107,78
JP 310 POKE DL+108,0
WS 320 POKE DL+109,80
YN 330 F$=F1$:HI=64:GOSUB 710
BX 340 F$=F2$:HI=96:GOSUB 710
BY 350 POKE PORTB,193+MODE+BANK1
ZU 360 F$=F4$:HI=64:GOSUB 710
CM 370 F$=F3$:HI=96:GOSUB 710
DH 380 POKE PORTB,193+MODE+BANK2
AJ 390 F$=F5$:HI=64:GOSUB 710
DA 400 F$=F6$:HI=96:GOSUB 710
DX 410 POKE PORTB,193+MODE+BANK3
AX 420 F$=F8$:HI=64:GOSUB 710
DP 430 F$=F7$:HI=96:GOSUB 710
AU 440 POKE PORTB,193+MODE+BANK0
UE 450 NUSCR=16384:HI=64:GOSUB 600
```

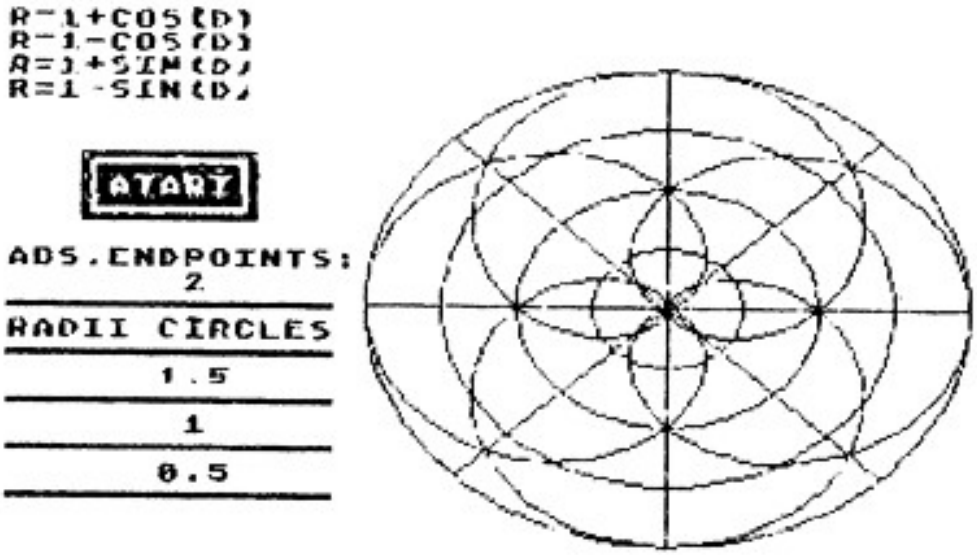
Listing 3 - pok.

```
AH 460 NUSCR=NUSCR+8192
VN 470 HI=96:GOSUB 600
CF 480 POKE PORTB,193+MODE+BANK1
TK 490 GOSUB 600
QY 500 NUSCR=16384
TM 510 HI=64:GOSUB 600
CX 520 POKE PORTB,193+MODE+BANK2
SZ 530 GOSUB 600
AE 540 NUSCR=NUSCR+8192
VK 550 HI=96:GOSUB 600
EI 560 POKE PORTB,193+MODE+BANK3
TH 570 GOSUB 600
RO 580 NUSCR=16384
EP 590 HI=64:GOSUB 600:GOTO 440
FZ 600 REM SET COLOR REGISTERS
PX 610 POKE 712,PEEK(NUSCR+7680+16)
TK 620 POKE 708,PEEK(NUSCR+7681+16)
VA 630 POKE 709,PEEK(NUSCR+7682+16)
SL 640 POKE 710,PEEK(NUSCR+7683+16)
FO 650 POKE DL+5,HI
HQ 660 POKE DL+109,HI+16
WH 670 TIME=100:P=PADDLE(0)
VF 680 IF P<>228 THEN TIME=P*10
PA 690 FOR DELAY=1 TO M:NEXT DELAY
ZD 700 RETURN
JY 710 IO=848:REM FILE LOADER
VK 720 OPEN #1,4,0,F$
EK 730 POKE IO+2,7
JG 740 POKE IO+4,16
IU 750 POKE IO+5,HI
FX 760 POKE IO+8,4
JL 770 POKE IO+9,30
TR 780 X=USR(ADR("hhh■LV■"),16)
LV 790 CLOSE #1        x1  x2
ZE 800 RETURN
```

Pozn:

▮ x1 = * inverzně

▮ x2 = d inverzně



Polar Plotter II

RNDr. Petr Emanovský
Jiří Hrdlička

Kružnice, spirály, růžice, kardioidy, lemniskáty - jsou to exotické rostliny z daleких krajů? Ne, jsou to názvy některých fascinujících křivek, které mohou být vytvořeny pomocí polárních rovnic.

Teď máte jedinečnou šanci proniknout do této tajemné oblasti. Zkuste proto experimentovat s polárními rovnicemi, které tvoří samostatnou oblast geometrie. Nevadí, že nejste matematik a neznáte problematiku polárních rovnic. Pokusíme se ji vám vysvětlit, ale k vlastnímu použití programu ji někteří z vás téměř nebudou potřebovat. Program Polar Plotter II je totiž vhodný nejen pro matematiky, ale i pro ty, kteří rádi zkoumají a vytvářejí na základě svých představ různé křivky.

Co je to polární rovnice?

(K pochopení problematiky polárních rovnic vám snad pomůže názorná představa (viz obr. 1). Sledujte text a snažte se nalézt popisované věci na obrázku.)

Nakreslete kružnici k - to by vám nemělo dělat problémy. Nyní vyznačte střed kružnice (bod S) a vodorovně nakreslete poloměr ze středu směrem doprava. Zvolte jednotkový poloměr. Kdybychom se pokusili změřit obvod kružnice k ve stejném měřítku, zjistili bychom, že obvod má délku 2*PI*Poloměr, kde PI je známé Ludolfovo číslo (PI=3.14159). PI je vlastně délka oblouku (v radiánech), který představuje polovinu obvodu kružnice (s jednotkovým poloměrem).

Tedy vezmeme-li číselnou osu s jednotkou, která je rovna poloměru naší kružnice k, a tuto osu "obtočíme" kolem kružnice, zjistíme, že obvod kružnice k je roven asi 6.28319 naší

jednotky, neboli právě 2*PI. Zde na číselné ose představuje PI délku mezi body 3 a 4 a jednotky této osy nazýváme radiány. Vidíme, že vyjádření úhlu v radiánech není nijak komplikované, neboť velikost každého úhlu se dá vyjádřit reálným číslem na číselné ose.

Ale vraťme se zpět k naší kružnici. Náš vodorovný jednotkový poloměr nazveme polární osa. Směrem doprava můžeme prodloužit tuto osu až do nekonečna. Levému krajnímu bodu na polární ose budeme říkat pól (střed kružnice S).

Nyní si představme přímku nebo lépe paprsek vycházející z pólu. Tento paprsek se může otáčet kolem pólu podobně jako ručička u hodinek s tím rozdílem, že se bude otáčet proti směru hodinových ručiček. Uvažujme bod na tomto paprsku. Polohu tohoto bodu můžeme určit pomocí jeho polárních souřadnic (R,theta). R je vzdálenost mezi pólem a uvažovaným bodem paprsku. Theta značí velikost úhlu v radiánech, který svírá paprsek s polární osou. Zkusme si vyjádřit vztah závislosti vzdálenosti R na úhlu theta.

Tento vztah může mít například tvar: R=theta. Pak s rostoucí velikostí úhlu theta úměrně roste i vzdálenost R. Takže tato rovnice nám určuje spirálu, která se zvětšuje s rostoucím úhlem theta.

Vezmeme-li v úvahu, že R může nabývat i záporných hodnot a theta libovolné reálné hodnoty (nikoliv jen od 0 do 2*PI), můžeme potom uvažovat i takovéto rovnice:

```
R=COS(theta)
R=COS(theta)*SIN(theta)
```

nebo i tuto "bláznivinu"

```
R=2+SIN(2*theta+2)-ABS(COS(3*theta+7))
```

Všechny tyto a podobné rovnice nazýváme polární rovnice.

Polar Plotter II

Natypujte listing 1 - program PPII.BAS. Zkontrolujte správnost programu pomocí kontrolních kódů programu TYPO II (Atari zpravodaj Olomouc 3-4) a pokud je vše v pořádku, uložte program na magnetofonovou kazetu (CSAVE) nebo na disk (SAVE "D:PPII.BAS"). Teprve teď spusťte program příkazem RUN.

Jak pracovat s programem Polar Plotter II?

Program se nejdříve zeptá na počet polárních rovnic. Můžete nechat vykreslit až čtyři polární rovnice v jednom grafu - to znamená v jednotném měřítku. Potom jednoduše vložíte rovnice podle návodu na obrazovce. Napište R=(a dále funkci s proměnnou D, kde proměnná D nahrazuje symbol úhlu theta) a stisknete RETURN. Pokud máte více nachystaných rovnic, opakujete tento postup.

Program počítá hodnoty funkce na intervalu (0,2*PI). Na stejném intervalu pak vykreslí graf funkce. Při vkládání polárních rovnic do počítače jste omezeni možnostmi jazyku Atari BASIC. Proto pozor na dělení nulou, druhou odmocninu ze záporné hodnoty, správné použití závorek atd.

V dalším kroku se vás počítač zeptá na absolutní konečný bod - tj. maximální přípustnou hodnotu vzdálenosti R. Můžete si vybrat, zda zadáte vlastní hodnotu nebo necháte počítač, aby si patřičnou hodnotu vypočítal sám (tzv. autoscaling). Program neohlásí chybu, jestliže hodnota funkce překročí vámi zvolený

konečný bod. Pouze vykreslí graf polární rovnice na daném intervalu. Důvody, které nás vedly k možnosti vlastní volby konečného bodu, jsou dva:
1) potřebujete transfokaci (zaostření) funkce
2) chcete docílit vhodný efekt

Po zadání všech těchto údajů začne vlastní práce programu. Program Polar Plotter II pomalu vykreslí graf funkce nebo grafy funkcí na obrazovku. Až skončí, zazní zvukový signál, který vám oznámí, že si můžete vybrat další možnosti programu:
1) tlačítko OPTION
Když poprvé stisknete tlačítko OPTION, na obrazovce se vykreslí osový kříž a kružnice, která má poloměr roven absolutnímu konečnému bodu. Dále se na obrazovku vypíše nebo vypíší polární rovnice pro jednotlivé funkce a hodnota absolutního konečného bodu. Po druhém stisknutí tohoto tlačítka vykreslí počítač pomocný osový kříž a tři soustředné kružnice na grafu. Zároveň vypíše hodnoty poloměrů těchto kružnic.
2) tlačítko SELECT
Toto tlačítko je určeno pro spuštění podprogramu uživatele. Doporučujeme ho využít k ukládání grafů (obrázků v grafice 24) na disk nebo na kazetu. Rutin, které dovedou ukládat data obrázku, bylo již mnoho publikováno a uživatel si může vybrat to, co mu nejlépe vyhovuje.
3) tlačítko START
Platí totéž jako o tlačítku SELECT. Doporučujeme využít tlačítko START ke spuštění hardcopy obrazovky. Zde záleží hlavně na typu tiskárny.
4) klávesa ESC
Stiskem této klávesy vymažeme obrazovku a spustíme program od začátku, tentokrát však už bez inicializace.

Popis programu Polar Plotter II
(program breakdown)

řádky 10 - 90
- hlavička programu (REM)
řádek 100
- skok na řádek inicializace programu (viz 500)
řádky 110 - 170
- hlavní smyčka k vykreslování grafu polární rovnice nebo rovnic
- část této smyčky je použita také k vykreslení soustředných kružnic, které nám pomáhají při orientaci v grafu a které tvoří i jakési pozadí grafu
- dále jsou zde hodnoty proměnných R a D (theta) převáděny do kartézské (pravoúhlé) soustavy souřadnic, což vlastně slouží k zobrazení bodů na obrazovku (použit grafický mod 24 =8+16)
řádky 200 - 230
- obsah těchto řádků se vytváří až po spuštění programu příkazem RUN
- zadáváme sem až čtyři povolené polární rovnice, tedy na řádku 200 je první polární rovnice, na řádku 210 druhá, atd.
- každý tento řádek je ještě mimo polární rovnici doplněn instrukcí RETURN
řádky 300 - 310
- rutina automatického měřítka grafu

řádek 400
- zvukový signál, který indikuje ukončení vykreslování grafů
- program je nyní připraven k použití tlačítek START, SELECT a OPTION plus klávesy ESC
řádky 410 - 440
- smyčka ke zjištění stisku tlačítek START, SELECT a OPTION a kontroly klávesy ESC
řádky 500 - 520
- inicializace a uložení strojového jazyka do řetězcové proměnné Z$
řádky 530 - 540
- úvodní obrazovka
řádky 550 - 600
- místo určené k zadávání polárních rovnic
- bere určené polární rovnice a umisťuje tyto rovnice do příslušných programových řádků (viz 200-230) metodou vynuceného čtení
řádky 610 - 630
- volba absolutního konečného bodu - může být buď automatická (autoscaling) nebo ruční
- v případě automatické volby konečného bodu volá program rutinu automatického měřítka (viz 300-310)
řádek 640
- nastavení obrazovky do grafického režimu (GRAPHICS 24)
řádky 650 - 940
- tyto řádky spolu s pomocnými podprogramy zabezpečují vykreslení osy a pozadí grafu
- dále vypisují návěští pro graf, kde najdeme použité měřítko a zadané polární rovnice
řádky 950 - 980
- podprogram, který si zapamatuje námi zadané polární rovnice pro další zpracování a použití
- volá se během zadávání polárních rovnic (viz 550-600)
řádky 1000 - 1060
- tyto řádky obsahují data strojové rutiny (viz 500-520)
- strojová rutina se využívá k navrstvení textu přes grafický režim 24 (můžete použít i ve vlastních programech)

Poslední slovo

Pro ty z vás, kteří chtějí program vyzkoušet a neví jak, uvádíme několik pěkných příkladů z bohaté nabídky polárních rovnic. Vyzkoušejte si je a potom je zkuste kombinovat mezi sebou. Na obrazovce tak vzniknou zajímavé, někdy trochu fantastické, obrazce. Přejeme příjemnou zábavu a poučení s programem Polar Plotter II...

Tabulka 1:
Příklady polárních rovnic

```
R=D      R=-D      R=D*2      R=-D*2
     R=1+COS(D)      R=1-COS(D)
     R=1+SIN(D)      R=1-SIN(D)
    R=2+SIN(3*D+2)-ABS(COS(3*D)
   R=2+COS(2*D+2)-ABS(SIN(3*D-7)
R=SIN(3*D) R=2*SIN(3*D) R=3*SIN(3*D)
R=D      R=2*D      R=3*D      R=4*D
    R=2+COS(3*D)-ABS(SIN(4*D))
     R=COS(D)        R=1+COS(D)
     R=2+COS(D)      R=3+COS(D)
      R=COS(D)       R=-COS(D)
      R=SIN(D)       R=-SIN(D)
```

Literatura:

1) David Baker - Polar Plotter
   (časopis ANALOG 1987/7 a 8)
2) Ian Chadwick - Mapping The Atari

Listing 1 - prog.PPII.BAS

```
EB 10 REM --------------------
WP 20 REM I POLAR PLOTTER II I
EV 30 REM --------------------
WU 40 REM for all ATARI XL/XE
RD 50 REM (c) 1987 by Dave Bader
JF 60 REM ANALOG Computing No.56
SX 70 REM july/august 1987
DS 80 REM (p) 1988 by GIA Software
RI 90 REM Atari Zpravodaj Olomouc!
MU 100 GOTO 500
WH 110 XM=96/Z:FOR Q=1 TO NUM:T=Q
HS 120 B=0:FOR D=0 TO 6.3 STEP 0.075:GOSUB 190+T*10

QQ 130 IF ABS(R)>Z THEN B=0:GOTO 170
WR 140 R=R*XM:X=COS(D)*R:Y=SIN(D)*R*0.83
WW 150 IF B=0 THEN PLOT X+208,191-(80+Y):B=1
PZ 160 DRAWTO X+208,191-(80+Y)
XU 170 NEXT D:NEXT Q:GOTO 400
RL 200 REM Formula entered during RUN
RN 210 REM Formula entered during RUN
RP 220 REM Formula entered during RUN
RR 230 REM Formula entered during RUN
NG 300 ? :? "AUTO SCALING...":Z=0:FOR T=1
     TO NUM:FOR D=0 TO 6.3 STEP 0.075:GOSU
   B 190+T*10:IF R>Z THEN Z=R
```

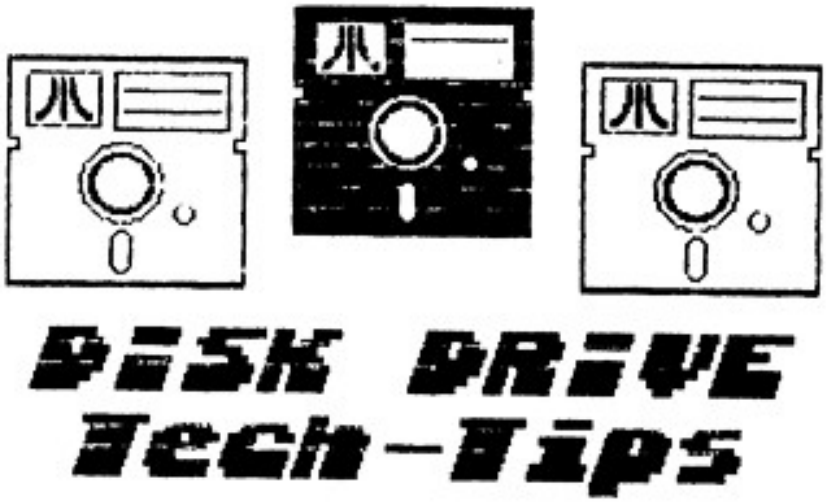
Listing 1 - pokr.

```
FB 310 NEXT D:NEXT T:GOTO 640
CQ 400 SOUND 0,100,10,10:FOR T=1 TO 100:N
   EXT T:SOUND 0,0,0,0:POKE 764,255
KA 410 T=PEEK(53279):IF T=6 THEN BD=BD+1: GOTO 650
GC 420 REM for LOAD/SAVE routine
VR 430 REM for HardCopy  routine
JS 440 IF PEEK(764)<>28 THEN 410
HU 450 BD=0:GOTO 530
SY 500 RAD :DIM Q$(49),W$(49),E$(49),R$(4
   9),T$(49),Z$(169):GRAPHICS 24
ZW 510 DM=PEEK(88)+PEEK(89)*256:DM=DM+40*191
VD 520 RESTORE 1000:FOR T=1 TO 168:READ Q
   :Z$(LEN(Z$)+1)=CHR$(Q):NEXT T
TW 530 GRAPHICS 0:POKE 82,0:POKE 710,144:
   POKE 709,12:POSITION 0,0:? "     DAVE
   BADERS  POLAR PLOTTER II        "
ZJ 540 ? "NUMBER OF FORMULA'S TO ENTER";:
   INPUT NUM:IF NUM>4 THEN 530
MD 550 FOR T=1 TO NUM
UA 560 ? :? "▌ FORMULA ENTRY SCREEN: ":?
   :? "INPUT FORMULA AS R=IN TERMS OF 'D'
   ,     WHERE 'D'=THETA"
AQ 570 ? "FORMULA # ";T:INPUT Q$
TO 580 GOSUB 900+T*10:POKE 559,0:Q$(LEN(Q
   $)+1)=":RETURN":? "▌":POSITION 0,6:? 1
   90+T*10;Q$:? :? "CONT":POSITION 0,0
UV 590 POKE 842,13:STOP
YD 600 POKE 842,12:POKE 559,34:NEXT T
KJ 610 ? :? "▌PLEASE CHOOSE [1] AUTOMATIC
    SCALING     (ABS. BOUNDS) [2] MANUAL
   SETTING.":INPUT T:IF T=1 THEN 630
BV 620 ? :? "ABSOLUTE ENDPOINTS";:INPUT Z
AJ 630 IF T=1 THEN GOSUB 300
AJ 640 GRAPHICS 24:POKE 710,58:POKE 712,5
   8:POKE 709,0:COLOR 1:GOTO 110
PZ 650 IF BD=1 THEN GOSUB 800:TRAP 400:T=
   51:GOTO 120
SL 660 IF BD=2 THEN PLOT 275,55:DRAWTO 14
   1,167:PLOT 141,55:DRAWTO 275,167:GOSUB
    860:FOR Q=1 TO 3:T=51+Q:GOTO 120
ND 670 GOTO 400
OX 700 R=Z:RETURN
UK 710 R=3*(Z/4):RETURN
KF 720 R=Z/2:RETURN
KZ 730 R=Z/4:RETURN
RB 800 FOR T=1 TO NUM:GOSUB 940+T*10:IF LEN (Q$)>39
       THEN Q$=Q$(1,39)
```

Listing 1 - pok.

```
DQ 810 X=0:Y=(T-1):GOSUB 840:NEXT T:PLOT
   112,111:DRAWTO 302,111:PLOT 208,32:DRA
   WTO 208,190
HU 820 X=3:Y=6:Q$="└──────┘":GOSUB 840:Y=7
   :Q$=" ATARI ":GOSUB 840:Y=8:Q$="└──────┘"
    :GOSUB 840
XS 830 X=0:Y=10:Q$="ABS.ENDPOINTS:":GOSUB
    840:Q$=STR$(Z):X=(14-LEN(Q$))/2:Y=11:
   GOSUB 840:RETURN
LA 840 Q=USR(ADR(Z$),X,Y,ADR(Q$),LEN(Q$))
   :RETURN
XQ 850 Q$="--------------":GOSUB 840:RETURN
DD 860 X=0:Y=12:GOSUB 850:Y=13:Q$="RADII
   CIRCLES":GOSUB 840:Y=14:GOSUB 850
XU 870 Q$=STR$(3*Z/4):Y=15:X=(14-LEN(Q$))
   /2:GOSUB 840:X=0:Y=16:GOSUB 850
LH 880 Q$=STR$(Z/2):Y=17:X=(14-LEN(Q$))/2
   :GOSUB 840:X=0:Y=18:GOSUB 850
YT 890 Q$=STR$(Z/4):Y=19:X=(14-LEN(Q$))/2
   :GOSUB 840:X=0:Y=20:GOSUB 850
ZF 900 RETURN
DF 910 W$=Q$:RETURN
ZV 920 E$=Q$:RETURN
CK 930 R$=Q$:RETURN
CW 940 T$=Q$:RETURN
EF 950 Q$=W$:RETURN
YT 960 Q$=E$:RETURN
CV 970 Q$=R$:RETURN
DN 980 Q$=T$:RETURN
UH 990 STOP
JR 1000 DATA 104,201,4,240,9,170,240,5,10
   4,104,202,208,251,96,104,133,215,104,1
   33,214,104,104,168,104,133
SQ 1010 DATA 217,104,133,216,104,104,240,
   236,133,212,24,165,214,101,88,133,214,
   165,89,101,215,133,215,152,240,15
ZS 1020 DATA 165,214,105,64,133,214,165,2
   15,105,1,133,215,136,208,241,132,221,1
   60,0,132,220,177,216,160,0,170
BV 1030 DATA 16,1,136,132,213,138,41,96,2
   08,4,169,64,16,14,201,32,208,4,169,0,1
   6,6,201,64,208,2
MM 1040 DATA 169,32,133,218,138,41,31,5,2
   18,133,218,169,0,162,3,6,218,42,202,20
   8,250,109,244,2,133,219
BZ 1050 DATA 164,221,177,218,69,213,164,2
   20,145,214,200,132,220,196,212,208,182
   ,24,165,214,105,40,133,214,144,2
CP 1060 DATA 230,215,230,221,169,8,197,22
   1,208,159,96,207,96
```

Pozn:Podtržené výrazy typovat inverzně.
└ nahrazuje ↖ (ESC SHRIFT CLEAR).



Tiskárna vypisuje program, vy pracujete na počítači. Vize budoucnosti? Ne, skutečnost na 8-bitových počítačích Atari XL/XE!

Backround Printer - Tiskárna v pozadí

Anselo Giambra

Protiví se mi čekání. Stávám se velmi netrpělivým, když tisknu výpis dlouhého programu. V zaměstnání je to jednoduché - na našem střediskovém počítačovém systému pokračuji v práci u terminálu nezávisle na tiskárně. Nebylo by pěkné, kdybych mohl dělat to samé doma na počítači Atari?

Teď už to dovedu. Program BP zavede do 8-bitového Atari novou rutinu pro obsluhu zařízení. Dá vám možnost pokračovat v práci na programu v BASICu - dokonce tento program spustit, zatímco tiskárna bude tisknout výpis programu (i jiného). Pokud vlastníte disketovou jednotkou a tiskárnu (nejlépe připojenou paralelně), program BP je určen pro vás.

Nastavení programu

Napište program - výpis 1. Použijte pomocný program TYPO II (Atari zpravodaj Olomouc 3-4/1988). Před spuštěním nahrejte program na disk. Potom vložte naformátovaný disk, na kterém je DOS 2.0 nebo DOS 2.5, do disketové jednotky a "odstartujte" program. Jestliže jste udělali nějakou chybu v DATA řádcích, program vám chybu ohlásí. Opravte ji, opět uložte program na disk a spusťte ho.

Program BP vytvoří na disku soubor AUTORUN.SYS. Proto nesmíte použít disk, který již tento soubor obsahuje.

Zapnutím počítače znovu obnovte systém - zároveň se

nahraje i soubor AUTORUN.SYS. Tento instaluje do nižší paměti počítače novou rutinu pro obsluhu nového zařízení. Toto zařízení je označeno symbolem B:.

Nahrejte do paměti počítače jakýkoliv BASICový program. Ujistěte se, zda je připojena a zapnuta tiskárna (případně i interface pro ni). Tiskárna by měla být "spřažena" přímo (on-line). Napište a odešlete: LIST "B:PRINT.SPL". Program BP nejdříve zapíše program na disk ve formě kódu ATASCII (pod názvem PRINT.SPL). Dále "zažehne" rutinu vertikálního přerušení (VBI). Ta začne tisknout soubor z disku na tiskárnu. V tomto okamžiku se na obrazovce objeví BASIC READY, a potom můžete pracovat s počítačem jako obvykle. Můžete program editovat (měnit, přidávat řádky atd.). Dokonce můžete nahrávat program do paměti, nebo ho ukládat na disk, případně můžete některé programy i spustit. Program BP pracuje vzhledem k tomu nezávisle.

Jak program BP pracuje? Je to jednoduché. Každou 1/60 sekundy, během VBI, přejde řízení do rutiny BP. Načte jeden sektor ze souboru PRINT.SPL do zásobníku v nižší paměti. V každém úspěšném přerušení, program pošle jeden znak do "sídla" tiskového manipulačního programu, který soustředí znaky do svého zásobníku. Když je tento zásobník plný, tiskový manipulační program ho pošle celý do tiskárny. Po každém vytištěném sektoru si program BP uloží do zásobníku následující sektor souboru PRINT.SPL. Až je soubor kompletně zpracován a vytištěn, řízení přejde do BASICu a rutina VBI se automaticky sama vypne.

Program BP dostává vaše 8-bitové Atari těsně na limit možné výkonnosti. Děláte s počítačem Atari dvě věci současně, na což nebyl vybaven. Když tedy spustíte program v BASICu, zpozorujete, že není tak rychlý. Může se stát, že se obrazovka na několik okamžiků doslova "zmrazí". To nastává při přenosu zásobníku přes I/O port.

Existují však i případy, které ovlivňují vaši práci s programem BP. Zásadně nesmíte používat program, který vyžaduje přístup k tiskárně. Také nejsou dovoleny příkazy LIST "P:" nebo LIST "B:". Pokud použijete příkaz LIST "B:", na obrazovce se objeví nápis PRINT.SPL IN USE (zařízení je v činnosti).

Také nevyměňujte disk, vložený do disketové jednotky 1, dříve, než program BP dočte soubor PRINT.SPL. (Ale můžete zapisovat a číst soubory z tohoto disku.)

Nemůžete však použít ani příkaz DOS. Je to způsobeno tím, že soubor DUP.SYS zábírá část paměti, kterou používá program BP. Jestliže použijete příkaz DOS, místo toho, abyste přešli do menu diskového operačního programu, rutina BP způsobí teplý start celého systému (odpovídá stisknutí klávesy SYSTEM RESET). Je jasné, že program BP potom přestává okamžitě fungovat.

Napsání příkazu DOS je možné tehdy, když program BP skončil s tiskem výpisu programu. V tom případě program BP zkontroluje, zda máte na disku soubor MEM.SAV. Pokud ano, DUP.SYS se uloží do paměti jako obvykle a když se vrátíte zpět do BASICu, program BP je stále připraven k použití. Jestliže na disku nemáte soubor MEM.SAV, program BP vynuluje všechny modifikované ukazovátka paměti (včetně LOMEM) a postoupí další činnost souboru DUP.SYS. Ale po návratu do BASICu zjistíte, že

program BP již nebude k dispozici.

Program BP bohužel nepracuje dokonale s tiskárnou připojenou sériově (nejhůře jsou na tom tiskárny "letter quality"). Tisknou pomalu a obrazovka se při tisku vždy "zmrazí". (Zkoušel jsem tiskárnu Atari 1029 a výsledek není tak špatný. Doporučuji vyzkoušet - pozn. překladatele)

Jsem zatím velice spokojený s programem BP a rutinou pro obsluhu nového zařízení. Myslím si, že budete také. Už nikdy žádné lelkování u počítače, zatímco tiskárna pracuje. Program BP vám umožní zvýšit produktivitu práce 8-bitového Atari XL/XE na vyšší míru než kdykoliv předtím.

Listing 2 - Assembly listing:

```
        .OPT NO LIST
;BACKGROUND PRINTER
;by Angelo Giambra
ADDR      =   $2200     ;START ADDRESS
DDEVIC    =   $0300     ;DEVICE
DUNIT     =   $0301     ;DEVICE
DCOMND    =   $0302     ;I/O COMMAND
DSTATS    =   $0303     ;DEVICE
DBUFLO    =   $0304     ;BUFFER ADDRESS
DTIMLO    =   $0306     ;BUFFER ADDRESS
DBYTLO    =   $0308     ;BUFFER ADDRESS
DBYTHI    =   $0309     ;BUFFER ADDRESS
DAUX1     =   $030A     ;SECTOR NUMBER
DSKINV    =   $E453     ;OS I/O ROUTINE
CIOV      =   $E456     ;OS I/O ROUTINE
NOTE      =   $26       ;NOTE SECTOR
READ      =   $04       ;OPEN FOR READ
PUTCHAR   =   $0B       ;PUT CHARACTERS
READCOM   =   $52       ;READ SECTOR CMND
STATUS    =   $0D       ;STATUS COMMAND
OPEN      =   $03       ;OPEN DEVICE
CLOSE     =   $0C       ;CLOSE DEVICE
OPENERROR =   129       ;CHANNEL OPEN
NOTFOUND  =   $AA       ;WORK FLAG
IOCB0     =   $00       ;CHANNEL 0
IOCB1     =   $10       ;CHANNEL 1
HATABS    =   $031A     ;HANDLER TABLE
ICCOM     =   $0342     ;I/O COMMAND
ICSTA     =   $0343     ;I/O STATUS
ICBAL     =   $0344     ;BUFFER ADDRESS
ICBLL     =   $0348     ;BUFFER LENGTH
ICAX1     =   $034A     ;AUXILIARY BYTE
ICAX3     =   $034C     ;AUXILIARY BYTE
```

```
DOSVEC     =    $0A        ;DUP SYS ADDRESS
DOSINI     =    $0C        ;DOS INIT
PBPNT      =    $1D
PBUFSZ     =    $1E
PRNBUF     =    $03C0
PTIMOT     =    $1C
LOMEM      =    $02E7      ;LOW MEMORY
WARMSTART  =    $E474
XITVBV     =    $E462      ;VBI EXIT
SETVB      =    $E45C      ;SET VBI ROUTINE
VVBLKD     =    $0224      ;VBI VECTOR ADDRESS
CR         =    $9B        ;CARRIAGE RETURN
COMMAND    =    $CB        ;TEMP STORAGE
SECTOR     =    $CC        ;SECTOR COUNT
INDEX      =    $CE        ;WORK INDEX
RETRY      =    $CF        ;RETRY FLAG
FLAG       =    $D0        ;WORK FLAG
BUSY       =    $D1        ;BUSY FLAG
;OFFSET TO RESIDENT PRINT HANDLER RETRY LOGIC
PRINTOFF   =    $EECB-$EEA7
@HANDLER TABLE
           *=   ADDR
           .WORD OPENIT-1
           .WORD CLOSEIT-1
           .WORD RETUEN-1
           .WORD WRITE-1
           .WORD RETURN-1
           .WORD RETURN-1
           JMP RETURN
DOS
           *=   *+2
DUPSAV
           *=   *+2
VBISAV
           *=   *+2
TIMER
           *=   *+1
;THE OPEN ROUTINE SUBSTITUES THE PRINT.SPL FILE NAME
;INTO THE IOCB, THEN CALLS THE DISK OPEN ROUTINE
OPENIT
           LDA SECTOR     ;PRINT.SPL IN USE?
           BEQ OPEN1      ;TEST OTHER BYTE
INUSE
           TXA            ;SAVE X REGISTER
           PHA
           LDX #IOCB0     ;CHANNEL 0-EDITOR
           LDA #PUTCHAR   ;WRITE CHARS
           STA ICCOM,X
           LDA # <MSG     ;IN USE MESSAGE
           STA ICBAL,X
           LDA # >MSG
           STA ICBAL+1,X
           LDA #17
           STA ICBLL,X    ;MESSAGE LENGTH
           LDA #0
```

```
            STA ICBLL+1,X
            JSR CIOV        ;WRITE IT
            PLA
            TAX             ;RESTORE X REGISTER
            LDY #OPENERROR
            TYA             ;GIVE ERROR MSG
            STA ICSTA,X     ;TO BASIC
            RTS
OPEN1
            LDA SECTOR+1
            BNE INUSE
            LDA # <DISKNAME
            STA ICBAL,X     ;SUBSTITUTE NAME
            LDA # >DISKNAME
            STA ICBAL+1,X
            LDA #1          ;SET COMMAND TO 1
            STA COMMAND
LOOKUP
            LDY #0
LOOK
            LDA HATABS,Y    ;FIND DISK HANDLER
            CMP # D         ;IS IT A  D ?
            BEQ FOUND       ;YES
            INY             ;NO, SKIP 2 BYTES
            INY
            INY
            BPL LOOK        ;LOOK AGAIN
FOUND
            INY
            LDA HATABS,Y    ;VECT TABLE ADDRESS
            STA INDEX       ;STORE IN INDEX
            INY
            LDA HATABS,Y
            STA INDEX+1
            LDY COMMAND     ;OFFSET TO OPEN ROUTLINE
            LDA (INDEX),Y   ;TRANSFER CNTRL
            PHA             ;WITH INDIRECT JP
            DEY
            LDA (INDEX),Y
            PHA             ;TARGET ADDRESS IS NOW ON THE STACK

            LDA FLAG        ;RESTORE THE CHAR SENT BY CIO
RETURN
            RTS             ;DO INDIRECT JUMP
WRITE
            STA FLAG        ;SAVE CHAR SENT BY CIO
            LDA #7          ;USED AS OFFSET TO WRITE ROUTINE
            STA COMMAND
            BNE LOOKUP
;THE CLOSE ROUTINE CLOSES THE PRINT.SPL FILE, REOPENS IT
;AND NOTES THE FIRST SECTOR. FINALLY IT CLOSES THE FILE
;AND FIRES UP THE VBI SERVICE ROUTINE
CLOSEIT
            LDA COMMAND     ;LAST CIO OP?
            CMP #7          ;A WRITE?
```

```
        BNE RETURN      ;NO, SO EXIT
        LDA #3          ;STORE OFFSET TO CLOSE ROUTINE
        STA COMMAND
        JSR LOOKUP      ;CALL DISK CLOSE ROUTINE
        LDX #IOCB1      ;CHANNEL 1
        LDA #OPEN       ;LOAD OPEN CMND
        STA ICCOM,X     ;STORE IN IOCB
        LDA # <DISKNAME
        STA ICBAL,X     ;STORE FILE NAME
        LDA # >DISKNAME
        STA ICBAL+1,X
        LDA #READ       ;OPEN INPUT
        STA ICAX1,X
        JSR CIOV        ;CALL OS I/O
        LDA #NOTE       ;NOTE SECTOR CMND
        STA ICCOM,X
        JSR CIOV        ;CALL I/O ROUTINE
        LDA ICAX3,X     ;GET THE SECTOR #
        STA SECTOR
        LDA ICAX3+1,X
        STA SECTOR+1
        LDA #CLOSE      ;CLOSE COMMAND
        STA ICCOM,X
        JSR CIOV        ;CLOSE THE FILE
        LDA #0          ;INIT WORK AREAS
        STA INDEX
        STA FLAG
        STA RETRY
        STA BUSY
        LDA # <WARMSTART
        STA DOSVEC      ;MAKE SURE CAN T USE DOS CMD
        LDA # >WARMSTART
        STA DOSVEC+1
        LDA #7          ;CALL AN IND VBI
        LDY # <VB       ;ADDR VBI ROUTINE
        LDX # >VB
        JSR SETVB       ;SET VBI
        RTS
OUT
        JMP XITVBV
VB
        LDA BUSY        ;DON T LET VBI
        BNE OUT         ;INTERRUPT ITSELF
        LDA FLAG        ;TIME TO PRINT?
        BNE PRINTER     ;YES
        STA COMMAND     ;INIT WORK AREA
        STA TIMER       ;INITIALIZE TIMER
        LDA #1
        STA DUNIT       ;DRIVE 1
        LDA #READCOM    ;READ COMMAND
        STA DCOMND
        LDA # <DATA     ;DATA BUFFER ADDRESS
        STA DBUFLO
        LDA # >DATA
        STA DBUFLO+1
```

```
            LDA SECTOR      ;LOAD SECTOR NO.
            STA DAUX1
            LDA SECTOR+1
            STA DAUX1+1
            JSR DSKINV      ;READ A SECTOR
            TYA             ;GET I/O STATUS
            BMI OUT         ;WAS IT BAD I/O?
            INC FLAG        ;SET FLAG
            BNE OUT         ;EXIT
PRINTER
            INC TIMER       ;INCREMENT TIMER
            LDA TIMER
            CMP #2          ;SLOW DOWN OUTPUT
            BNE OUT
            LDA #0          ;RESET TIMER
            STA TIMER
            LDA RETRY       ;LAST I/O BAD?
            BNE TRYAGAIN    ;YES, TRY AGAIN
            INC BUSY        ;SET BUSY FLAG
            LDY INDEX       ;CHARACTER INDEX
            LDA DATA,Y      ;LOAD A CHARACTER
;THE FOLLOWING ROUTINE EMULATES THE OPERATING SYSTEM
;PRINT HANDLER WRITE ROUTINE
PRINT
            PHA             ;SAVE CHARACTER
            LDY #$57        ;WRITE COMMAND
            LDX #$28        ;BUFFER SIZE
            LDA #$4E        ;NORMAL MODE
            STA DAUX1
            STY DCOMND
            STX PBUFSZ
            LDX PBPNT
            PLA             ;GET CHARACTER
            STA PRNBUF,X    ;PUT IN BUFFER
            INX             ;INC BUFF POINTER
            CPX PBUFSZ      ;= BUFFER SIZE?
            BEQ BUFFUL
            STX PBPNT       ;SAVE POINTER
            CMP #$98        ;EOL?
            BEQ BLFILL      ;YES, BLANK FILL
            LDY #1          ;GOOD STATUS
            JMP IN
BLFILL
            LDA #$20        ;LOAD A SPACE
FILLBF
            STA PRNBUF,X
            INX
            CPX PBUFSZ      ;BUFFER FULL?
            BNE FILLBF
BUFFUL
            LDA #0          ;CLEAR BUFFER
            STA PBPNT       ;POINTER
            LDX #$C0        ;SET BUFF POINTER
            LDY #$03
            STX DBUFLO
```

```
            STY DBUFLO+1
            LDA #$40        ;STORE DEVICE
            STA DDEVIC
            LDA #1
            STA DUNIT
            LDA #$80        ;DIRECTION = OUTPUT
            STA DSTATS
            LDA PBUFSZ      ;STORE BUFF SIZE
            STA DBYTLO
            LDA #0
            STA DBYTHI
            LDA PTIMOT      ;TIMEOUT VALUE
            STA DTIMLO
            JSR $E459
IN
            DEC BUSY        ;BUST FLAG OFF
            TYA             ;GET STATUS
            BPL AHEAD       ;GOOD I/O
            INC RETRY       ;SET RETRY FLAG
            BNE JP
AHEAD
            LDY INDEX
            INY             ;INC CHAR INDEX
            CPY DATA+127    ;END OF SECTOR?
            BCS NEXTSECTOR;YES
            STY INDEX       ;NO, SAVE IT
JP
            JMP OUT
TRYAGAIN
            DEC RETRY       ;RETRY FLAG OFF
            INC BUSY        ;SET BUSY FLAG
RETADDR
            JMP BUFFUL      ;RETRY WRITING THE BUFFER
EXITVB
            JSR $00         ;PRINT HANDLER CLOSE ROUTINE
            LDA VBISAV      ;RESTORE VBI VECT
            STA VVBLKD
            LDA VBISAV+1
            STA VVBLKD+1
            LDA # <DUPEXIT;RESTORE DOSVEC
            STA DOSVEC
            LDA # >DUPEXIT
            STA DOSVEC+1
            BNE JP
NEXTSECTOR
            LDA #0          ;RESET WORK AREAS
            STA INDEX
            STA FLAG
            LDA DATA+125    ;GET NEXT SECTOR
            AND #$03        ;GET HIGH BYTE
            STA SECTOR+1
            BEQ NEXT1
            INC COMMAND     ;THERE IS MORE
NEXT1
            LDA DATA+126    ;GET LOW BYTE
```

```
            STA SECTOR
            BEQ NEXT2
            INC COMMAND    ;THERE IS MORE
NEXT2
            LDA COMMAND    ;MORE SECTORS?
            BEQ EXITVB     ;NO, END VBI
            JMP OUT        ;EXIT VBI
DISKNAME    .BYTE "D:PRINT.SPL"
MEMSAV
            .BYTE "D:MEM.SAV"
MSG
            .BYTE "PRINT.SPL IN USE",CR
DATA                       ;DISK DATA BUFFER
            *= *+128
DUPEXIT
            LDX #IOCB1     ;CHANNEL 1
            LDA #STATUS    ;GET STATUS OF MEM.SAV FILE
            STA ICCOM,X
            LDA # <MEMSAV
            STA ICBAL,X
            LDA # >MEMSAV
            STA ICBAL+1,X
            JSR CIOV
            CPY #NOTFOUND ;IS IT THERE?
            BNE JUMPDUP    ;YES, GET OUT
            LDY #0
FINDB
            LDA HATABS,Y   ;FIND B DEVICE
            CMP # B
            BEQ FOUNDB     ;FOUND IT
            INY
            INY
            INY
            BPL FINDB
FOUNDB
            LDA #0         ;TAKE OUT OF TABL
            STA HATABS,Y
            LDA # <ADDR    ;RESTORE LOMEM TO ORIGINAL VALUE
            STA LOMEM
            LDA # >ADDR
            STA LOMEM+1
            LDA DOS        ;RESTORE DOSINI TO ORIGINAL VALUE
            STA DOSINI
            LDA DOS+1
            STA DOSINI+1
            LDA DUPSAV     ;RESTORE DOSVEC TO ORIGINAL VALUE
            STA DOSVEC
            LDA DUPSAV+1
            STA DOSVEC+1
            JMP (DOSVEC)   ;JUMP TO DUP.SYS
JUMPDUP
            LDA DUPSAV
            STA $00
            LDA DUPSAV+1
            STA $01
```

```
        JMP ($00)
RESET
        LDA #0          ;RESET SECTOR NUMBER
        STA SECTOR
        STA SECTOR+1
        LDA DOS         ;GET DOSINI VECT
        STA DOSADDR+1   ;CREATE JSR ADDRESS
        LDA DOS+1
        STA DOSADDR+2
DOSADDR
        JSR $00         ;DOS INIT
        JMP INITMORE    ;REINIT HANDLER TABLE
INIT
        LDA DOSINI      ;GET DOSINI VECT AND SAVE IT
        STA DOS
        LDA DOSINI+1
        STA DOS+1
        LDA # <RESET    ;POINT DOSINI TO OUR RESET LOGIC
        STA DOSINI
        LDA # >RESET
        STA DOSINI+1
INITMORE
        LDA DOSVEC      ;SAVE DOSVEC
        STA DUPSAV      ;IN DUPSAV
        LDA DOSVEC+1
        STA DUPSAV+1
        LDA # <DUPEXIT;POINT DOSVEC TO DUPEXIT
        STA DOSVEC
        LDA # >DUPEXIT
        STA DOSVEC+1
        LDA VVBLKD      ;SAVE ORIG VBI VECTOR
        STA VBISAV
        LDA VVBLKD+1
        STA VBISAV+1
        LDA # <FINI     ;SET LOMEM TO POINT PAST END
        STA LOMEM       ;OF THIS PROGRAM
        LDA # >FINI
        STA LOMEM+1
        LDY #0
SEEK
        LDA HATABS,Y    ;LOOK IN HANDLER TABLE
        BEQ PUTIT       ;FOR A FREE SPACE
        CMP # P         ;IS IT THE PRINTER HANDLER?
        BNE PUT1
        JSR GETIT       ;YES, PROCESS IT
PUT1
        INY             ;SKIP 2 BYTES
        INY
        INY
        BPL SEEK
PUTIT
        LDA # B         ;HANDLER NAME IN TABLE
        STA HATABS,Y
        INY
        LDA # <ADDR     ;STORE ADDRESS OF OUR VECT TABLE
```

```
        STA HATABS,Y
        INY
        LDA # >ADDR
        STA HATABS,Y
        RTS             ;EXIT
GETIT
        TYA             ;SAVE THE Y REG ON THE STACK
        PHA
        INY
        LDA HATABS,Y    ;GET VECT TABLE ADDRESS
        STA INDEX
        INY
        LDA HATABS,Y
        STA INDEX+1
        LDY #2          ;OFFSET TO CLOSE ROUTINE
        CLC
        LDA (INDEX),Y   ;LOAD CLOSE ADDRESS
        ADC #1          ;ADD 1 TO IT
        STA EXITVB+1    ;MODIFY JSR CODE
        INY
        LDA (INDEX),Y
        ADC #0
        STA EXITVB+2
        PLA             ;RESTORE Y INDEX
        TAY
        RTS
FINI
        *= $02E0
        .WORD INIT
```

Listing - prog.BGPRINT.BAS

```
QN 10 REM ----------------------
GL 20 REM | BACKGROUND PRINTER |
RD 30 REM ----------------------
DO 40 REM by Angelo Giambra
PH 50 REM (c) 1987 ANALOG Computing
DQ 60 REM (p) 1988 by GIA Software
WX 70 REM for all ATARI XL/XE
JQ 80 REM program BGPRINT.BAS requires:
ER 90 REM disk drive & printer
MA 100 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0
   ,0,0,10,11,12,13,14,15
QI 110 DIM DAT$(96),HEX(22):FOR X=0 TO 22
   :READ N:HEX(X)=N:NEXT X:LINE=160:RESTO
   RE 500:TRAP 190:? ")CHECKING DATA"
WK 120 TOTAL=0:LINE=LINE+10:POSITION 2,2:
   ? "LINE:";LINE:READ DAT$:IF LEN(DAT$)<
   >96 THEN 220
TU 130 DATLIN=PEEK(183)+PEEK(184)*256:IF
   DATLIN<>LINE THEN ? "LINE ";LINE;" MIS
   SING!":END
JV 140 FOR X=1 TO LEN(DAT$)-1 STEP 2:D1=A
   SC(DAT$(X,X))-48:D2=ASC(DAT$(X+1,X+1))
   -48:BYTE=HEX(D1)*16+HEX(D2)
QE 150 IF PASS=2 THEN PUT #1,BYTE:NEXT X:
   READ CHKSUM:GOTO 120
DN 160 TOTAL=TOTAL+HEX(D1)+HEX(D2):NEXT X
TL 170 READ CHKSUM:IF TOTAL=CHKSUM THEN 120
NC 180 GOTO 220
LU 190 IF PEEK(195)<>6 AND PEEK(195)<>5 T
   HEN 220
NS 200 IF PASS=0 THEN OPEN #1,8,0,"D:AUTO
   RUN.SYS":PASS=2:LINE=160:RESTORE 500:T
   RAP 210:? ")CREATING FILE":GOTO 120
IA 210 CLOSE #1:END
WJ 220 IF LEN(DAT$)=30 AND LINE=330 THEN
   TRAP 200:GOTO 130
MK 230 ? "BAD DATA: LINE ";LINE:END
LM 500 DATA FFFF00220E2215228222792 27A227
   92279224C7A2216221123A5CCF0298A48A200A
   90B9D4203A9FA9D4403A9239D4503,545
TE 510 DATA A9119D4803A9009D49032056E468A
   AA081989D430360A5CDD0D3A9E69D4403A9239
   D4503A90185CBA000B91A03C944F0,597
DQ 520 DATA 05C8C8C810F4C8B91A0385CEC8B91
   A0385CFA4CBB1CE4888B1CE48A5D06085D0A90
   785CBD0D2A5CBC907D0F1A90385CB,728
```

Listing - BGPRINT.BAS, pokr.

```
KG 530 DATA 205522A210A9039D4203A9E69D440
   3A9239D4503A9049D4A032056E4A9269D42032
   056E4BD4C0385CCBD4D0385CDA90C,587
OI 540 DATA 9D42032056E4A90085CE85D085CF8
   5D1A974850AA9E4850BA907A0E2A222205CE46
   04C62E4A5D1D0F9A5D0D02D85CB8D,659
XL 550 DATA 1522A9018D0103A9528D0203A90B8
   D0403A9248D0503A5CC8D0A03A5CD8D0B03205
   3E4983012230A24CCE6D0D0C8EE15,563
BP 560 DATA 22AD1522C902D0BEA9008D1522A5C
   FD07BE6D1A4CEB90B2448A057A228A94E8D0A0
   38C0203861EA61D689DC003E8E41E,649
SR 570 DATA F015861DC99BF005A0014C8F23A92
   09DC003E8E41ED0F8A900851DA2C0A0038E040
   38C0503A9408D0003A9018D0103A9,565
LC 580 DATA 808D0303A51E8D0803A9008D0903A
   51C8D06032059E4C6D1981004E6CFD00AA4CEC
   8CC8A24B02584CE4CDF22C6CFE6D1,656
FE 590 DATA 4C6023200000AD13228D2402AD142
   28D2502A98B850AA924850BD0DDA90085CE85D
   0AD8824290385CDF002E6CBAD8924,587
NP 600 DATA 85CCF002E6CBA5CBF0C94CDF22443
   A5052494E542E53504C443A4D454D2E534156D
   0D2C9CED4AED3D0CCA0C9CEA0D5D3,711
CM 610 DATA C59B8B247D25A210A90D9D4203A9F
   19D4403A9239D45032056E4C0AAD034A000B91
   A03C942F005C8C8C810F4A900991A,593
MI 620 DATA 03A9008DE702A9228DE802AD0F228
   50CAD1022850DAD1122850AAD122285B6C0A0
   0AD11228500AD122285016C0000A9,512
FZ 630 DATA 0085CC85CDAD0F228DF724AD10228
   DF8242000004C0E25A50C8D0F22A50D8D1022A
   9E4850CA924850DA50A8D1122A50B,595
QT 640 DATA 8D1222A98B850AA924850BAD24028
   D1322AD25028D1422A97E8DE702A9258DE802A
   000B91A03F00CC950D003205B25C8,573
KA 650 DATA C8C810EFA942991A03C8A900991A0
   3C8A922991A03609848C8B91A0385CEC8B91A0
   385CFA00218B1CE69018DAD23C8B1,651
NK 660 DATA CE69008DAE2368A860E002E102FC2
   4,195
```

Všechny proměnné vyhledává rychlostí blesku. Možnost výpisu na obrazovku nebo na tiskárnu. Určeno pro všechny počítače Atari XL/XE.

Variable Searcher (VS) - Vyhledávač proměnných

Steven Anderson

Všechny programy v BASICu - nebo alespoň 99% z nich - používají proměnné. Kontrolovat je, prohlížet nebo měnit pomocí editačních možností počítačů Atari je většinou obtížná a nudná práce vyžadující trpělivost.

Jedna z ukrytých vlastností 8-bitových počítačů Atari je jejich schopnost používat dlouhé názvy pro proměnné. Někteří programátoři rádi mění názvy proměnných použitých v programu na delší, ale pro ně srozumitelnější názvy; jiní naopak chtějí vytvořit z dlouhých názvů kratší. Program VS může najít a vzájemně odkázat všechny názvy proměnných v několika sekundách. Podle vašeho přání je pak vytiskne buď na obrazovku nebo na tiskárnu. Vypíše vždy nejen název proměnné, ale i čísla řádků, na kterých můžeme tuto proměnnou nalézt. Vypíše i ty proměnné, které jste zadali v přímém počítačovém módu.

VS je program napsaný ve strojovém jazyku. Uloží se do paměti pomocí krátkého zaváděcího programu v BASICu. Zabírá šestou stránku paměti - od adresy 1536 po 1791 (\$0600 - \$06FF), což je nejpoužívanější část paměti pro krátké programy napsané ve strojovém kódu. Sem BASIC nezasahuje a klávesa SYSTEM RESET nemá nežádoucí vliv.

Abyste mohli spustit program VS, musíte napsat v přímém módu: X=USR (1536). Pokud chcete vypsat seznam proměnných na tiskárnu, potom před tento příkaz zadejte: POKE 203,86. Pro obnovení výstupu na obrazovku stačí napsat: POKE 203,69.

Program VS nejdříve najde tabulku názvů proměnných. Adresu začátku této tabulky můžete najít v paměťových adresách 130 a 131. Nalézt tuto hodnotu je v BASICu jednoduché - stačí napsat PRINT PEEK(130)+256*PEEK(131).

Konečná adresa tabulky se nachází v paměťových lokacích 132 a 133. Když je tato hodnota zjištěna, program VS vyhledá a vypíše každý název proměnné, uloží její číselné znamení. Potom pro toto znamení prohledává tu oblast paměti, do které jsou ukládány programy v BASICu. Jestliže program najde v paměti číselné znamení odpovídající dané proměnné, zkontroluje, zda není použito v příkazech REM, DATA nebo PRINT. Data obsažená v těchto příkazech totiž mohou mít tytéž hodnoty jako znamení proměnné - ale nejsou to proměnné! Po této kontrole program VS vypíše číslo řádku, kde byla proměnná nalezena. Tento proces se neustále opakuje až do vyčerpání hodnot tabulky názvů proměnných. Potom program končí a vrací se automaticky zpět do BASICu.

Program VS nekontroluje opakování proměnné - jestliže je proměnná nalezena několikrát v tomtéž řádku, potom je několikrát vypsáno i číslo tohoto řádku.

Všechny zadané proměnné jsou uloženy v tabulce - dokonce i ty, které byly zadány v přímém počítačovém módu. Poznáme je velice snadno, chybí jim totiž čísla řádků.

Napište zaváděcí program v BASICu - výpis 1. Ten po spuštění vytvoří strojový kód programu VS. Buďte extrémně opatrní při zadávání datových řádků - použijte program TYPO II pro správné zadání typovaných řádků (Atari zpravodaj Olomouc 3/4 1988). Před spuštěním programu VS si uložte kopii tohoto programu na disk nebo na msf. kazetu. Potom teprve spusťte BASICový zaváděč. Jestliže se objeví na obrazovce nápis READY, je všechno v pořádku - program VS je na svém místě a připraven k použití.

Otestujte program na zaváděči, abyste zjistili, zda správně pracuje. Pokud chcete použít program VS u jiného programu, napište NEW a tento program nahrejte do paměti. Ke spuštění programu VS napište: X=USR (1536).

Jestliže váš BASICový program potřebuje jakoukoliv část paměti mezi adresami 1536 až 1791, nespouštějte ho, dokud nepoužijete program VS. Změna v této paměťové oblasti má za důsledek smazání programu VS, v určitých případech i zhroucení operačního systému.

U dlouhého programu, který má mnoho proměnných, se nemusí vlézt všechno na jednu obrazovku. Většinou dochází k posuvu řádků. Proto používejte kombinace kláves CONTROL+1 k přerušení a obnovení výpisu.

Program VS automaticky nastavuje levý okraj obrazovky na nulu. To umožňuje úhledný výpis na obrazovku. Ale nenastavuje funkci tabulátoru (TAB). Pozměnění hodnoty tabulátoru uživatelem může způsobit zmatek ve výstupu dat, proto ho doporučuji nevyužívat.

Program VS pracuje podobně jako příkazy LVAR v jazycích BASIC XL a XE nebo DUMP v jazyku TURBO BASIC XL. Lze ho použít na každém 8-bitovém počítači Atari XL/XE s nebo bez DOSu. Program pracuje velmi dobře s většinou verzí DOSu, nejlépe však se systémem DOS 2.0 nebo DOS 2.5.

Listing prog.VSEARCH.BAS

```
WY 10 REM ---------------------
RI 20 REM | VARIABLE SEARCHER |
XQ 30 REM ---------------------
PX 40 REM by Steven Anderson
PH 50 REM (c) 1987 ANALOG Computing
DQ 60 REM (p) by GIA Software
WX 70 REM for all ATARI XL/XE
FH 80 REM type: X=USR(1536) to activate
PZ 90 REM program uses page 6 of memory
JN 100 FOR I=1536 TO 1791:READ D:POKE I,D:NEXT I
PH 110 POKE 203,69
OS 120 POKE 204,58
QW 130 POKE 205,88
SN 140 POKE 206,155
OA 150 END
EB 1000 DATA 165,130,133,0,165,131,133,1,
    169,127,133,209,162,48,169,203
DJ 1010 DATA 157,68,3,104,133,82,157,69,3
   ,157,75,3,169,8,157,74
VO 1020 DATA 3,169,3,157,66,3,32,86,228,1
   60,0,177,0,16,16,41
IH 1030 DATA 127,32,93,6,32,91,6,230,209,
   32,111,6,32,91,6,32
OL 1040 DATA 93,6,230,0,208,2,230,1,165,0
   ,197,132,208,219,165,1
IG 1050 DATA 197,133,208,213,169,12,141,1
   14,3,208,15,169,155,162,11,142
EB 1060 DATA 114,3,162,0,142,72,3,142,73,
   3,162,48,76,86,228,165
YM 1070 DATA 136,133,2,165,137,133,3,32,2
   44,6,133,207,32,244,6,133
TO 1080 DATA 208,201,128,176,121,32,244,6
   ,32,244,6,32,244,6,201,2
AD 1090 DATA 176,9,32,244,6,201,155,208,2
   49,240,220,32,244,6,201,20
KI 1100 DATA 240,230,201,27,240,226,201,2
   2,240,205,201,14,208,10,162,6
KV 1110 DATA 32,244,6,202,208,250,240,227
   ,201,15,208,6,32,244,6,170
ZD 1120 DATA 208,238,197,209,208,213,165,
   207,133,212,165,208,133,213,32,170
GZ 1130 DATA 217,32,230,216,160,255,132,1
   75,230,175,164,175,192,8,240,187
HG 1140 DATA 177,243,72,41,127,32,93,6,10
   4,16,237,169,160,164,175,200
SI 1150 DATA 145,243,208,228,160,0,177,2,
   230,2,208,2,230,3,96,0
RD 1160 DATA 230,2,208,2,230,3,96,0
```

Klaviatura Shakuhachi

Albert Baggetta

Jedna z největších vymožeností domácích počítačů je jejich schopnost simulace. Počítač může graficky modelovat objekty, logicky napodobovat reálné situace, dokáže napodobovat opravdu celý svět zvuků a tisíce z nich věrně reprodukovat.

Jako hudebník jsem nadšen schopností imitovat hudební zvuky u počítačů Atari. Kdykoliv slyším zajímavý nebo jedinečný zvuk, zapnu svůj počítač Atari 130 XE a zkusím zvuk zaznamenat a reprodukovat. Když počítači trochu pomůžu, většinou to jde. Nejdřív si zjistím, odkud zvuk přichází, čím a jak je reprodukován. Je to "trhaný" zvuk? Ztrácí se rychle nebo pomalu? Je zvukový efekt vytvářen jedním nebo více hlasy? Na tyto a podobné otázky musím najít odpověď (Rozhodnout není vždy jednoduché!). Potom zasednu za staré dobré 8-bitové Atari a zkusíme spolu tento zvuk napodobit.

Nedávno jsem viděl film "The Karate Kid". Zaujal mě jeho hudební doprovod - zvláště hra na japonský nástroj Sho nebo Shakuhachi, což jsou flétny vyrobené ze stvolů bambusu. Tento zvuk je nám známý především z filmů nebo televize a mně se zdál velice vhodný a zajímavý pro počítačovou simulaci. Po mnoha experimentech se mi podařilo vytvořit doprovodný program - Klaviaturu Shakuhachi.

Rozhodl jsem se, že zvuk bude mít dvě charakteristiky. Hudební tón je podmalován přídechem - závany větru. Kvůli tomu přechází hlasitost tónu od minima přes maximum až do nulové hodnoty (zastavení přídechu). Jestliže znáte něco o technice reprodukování zvuků na počítačích Atari, víte, že jsem potřeboval nějakou vysokou intonaci tzv. "bílého zvuku". Ten spolu s hudebním tónem, který zvolna narůstá až k svému maximu hlasitosti, tvoří výsledný efekt. Vše je vytvořeno krátkou smyčkou a příkazem SOUND na řádcích 300-310 programu.

Potřeboval jsem také znát některé používané noty. Malé zkoumání v knihovně odhalilo, že Japonci sice nepoužívají evropské hudební měřítko, ale většina jejich not je shodná s naší normou. Typický příklad následuje:

Nota	Intonace
G	162
A	144
C	121
D	108
E	96
G	81

Můj program náhodně vybírá kombinace těchto not k vytvoření typicky japonské melodie. Náhodné pomlky pomáhají simulovat časování a tvůrčí originalitu. (Ještě někdo tvrdí, že počítače nemohou být zároveň umělci?)

V programu jsem použil grafický mód 0 s textovým okénkem. Nahoře na obrazovce se objeví název programu a japonský motiv páru plachtících ptáků hnaných větrem proti slunečné obloze.

Listing prog.SHAKEY.BAS

```
KU 10 REM ------------------------
CJ 20 REM I SHAKUHACHI KEYBOARD I
LI 30 REM ------------------------
IG 40 REM by Albert Bassetta
PH 50 REM (c) 1987 ANALOG Computing
DQ 60 REM (p) 1988 by GIA Software
YH 70 REM program SHAKEY.BAS
WY 80 REM for all ATARI XL/XE
BM 90 REM disk or cassette
IQ 100 GRAPHICS 0:POKE 703,4:POKE 752,1:?
    #6;CHR$(125):SETCOLOR 1,0,0:SETCOLOR
   2,0,0:SETCOLOR 4,0,0
KH 110 ? #6:? #6:? #6:? #6
AW 120 ? #6;"          ";CHR$(8);CHR$(10)
FZ 130 ? #6;"        ";CHR$(8);CHR$(7);CHR
   $(136);CHR$(138)
AW 140 ? #6;"        ";CHR$(136)
WD 150 ? #6;"      Shakuhachi  Keyboard"
MV 160 ? #6:? #6;"
    ";CHR$(8);CHR$(10)
IH 170 ? #6;"                        ";CHR$
   (8);CHR$(7);CHR$(136);CHR$(138)
UF 180 ? #6;"                        ";CHR$
   (136)
NI 190 POSITION 0,18:? #6;"             JA
   PANESE SCALE"
NI 200 POSITION 0,19:? #6;"           ^^^^
   ^^^^^^^^^^^^^^^"
TV 210 FOR I=1 TO 15
XQ 220 SETCOLOR 2,1,I:SETCOLOR 4,1,I:SETC
   OLOR 1,1,15-I:NEXT I
OB 230 X=INT(RND(0)*6)+1
FI 240 ON X GOTO 250,260,270,280,290,300
TZ 250 P=162:? "G ";:GOTO 310
QX 260 P=144:? "A ";:GOTO 310
QK 270 P=121:? "C ";:GOTO 310
SV 280 P=108:? "D ";:GOTO 310
UF 290 P=96:? "E ";:GOTO 310
ST 300 P=81:? "G ";:GOTO 310
MJ 310 FOR D=0 TO 4 STEP 0.3:SOUND 0,P,10
   ,D:SOUND 1,1,8,1:NEXT D
FP 320 SOUND 0,P,10,D:SOUND 1,1,8,1
QD 330 FOR E=1 TO INT(RND(0)*350):NEXT E
CX 340 SOUND 1,0,0,0:SOUND 0,0,0,0:REM RE
   MOVE REM FOR DELAY BETWEEN NOTES AB=1^ 1
NJ 350 GOTO 230
```

V dolní části obrazovky se budou zobrazovat hrané noty pro vaši lepší orientaci.

Napište program Klaviatura Shakuhachi a uložte ho na msf. kazetu nebo na disk. Při typování použijte pomocný program TYPO II pro bezchybnost vašeho programu (viz Atari zpravodaj Olomouc 3-4/1988). Potom vám nezbude už nic jiného, než si obléct kimono, sednout si k šálku pravého japonského čaje a zaposlouchat se do orientálních zvuků synthezátoru Shakuhachi

(c) 1988 by GIA Software
Jiří Hrdlička
ATARI 130 XE

Dok.programu ze str.57

Listing prog.VRTULNIK - pokr.

```
PZ 2040 DATA 0,254,0,0,128,192,96,32,16,1
   76,224,192,32,16,56,0
YD 2050 DATA 0,127,0,64,64,65,127,240,158
   ,134,131,3,2,4,63,0
XE 2060 DATA 0,224,128,128,240,136,132,14
   0,254,14,252,248,8,5,254,0
OO 2100 STOP
```

Pozor !!

Řádek 215 = (.....):
h = malé písmeno h
" = uvozovky inverzně
/ = CTRL+F zn.srdce
- = mezera inverzně
\ = CTRL+, (zn.srdce)
) = zavorka inverzně
= CTRL+G
L = velké písmeno L
\ = SHIFT++ (zn.opačného lomítka)
d = malé písmeno d inverzně

Fort Apocalypse

Synapse Software
Machine Language - 32 kB

Z dobře střežené pevnosti, která se rozkládá v podzemí, je nutno zachránit 16 rukojmí. Tato pevnost se nazývá Fort Apocalypse.

Hra je určena pro všechny 8-bitové počítače Atari řady XL/XE minimálně s 32 KB RAM. Je napsána ve 100 procentním strojovém jazyku a byla vyrobena firmou Synapse Software.

Z dobře střežené podzemní pevnosti, která se nazývá Fort Apocalypse, je nutné zachránit šestnáct rukojmích. K dispozici máte joystickem ovládaný vrtulník. Tato útočící zbraň může odpalovat rakety (vrtulník je otočen bokem) nebo bombardovat objekty plazmovými bombami (vrtulník je vidět zepředu).

Pevnost je dokonale střežena ze všech stran. Kralthanovy tanky vypalují samonaváděcí střely, které sledují pohyb vaší helikoptéry, a ve vzduchu patrolují vrtulníky ovládané roboty. Naštěstí můžete sledovat jejich manévry na obrazovce radaru NAVATRON, který je umístěn nahoře na obrazovce. Tento miniaturní radar sleduje okolí vašeho vrtulníku při pohybu po celé pevnosti.

Než se pustíte do záchrany rukojmích, je třeba vrtulník natankovat. K tomu slouží plošina označená nápisem FUEL, česky palivo. Vstup do pevnosti je spojen s prvními nesnázemi. Je třeba vynutit si vstup bombardováním plazmou a přitom neustále sledovat pohyb nepřátelských sil na radaru a ničit je.

Jakmile se dostanete dovnitř pevnosti, musíte opatrně proletět úzkými chodbami, kde často zažijete nepříjemné překvapení ve formě ostřelování vaší záchranné helikoptéry buď tanky, roboty nebo létacími minami. Chodby pevnosti jsou velice úzké a lehký dotyk zdi a vrtulníku znamená začít hru znovu. Přístupové dveře k laserovým a hyperprostorovým komorám se dají otevřít raketami. Nejkritičtější úsek první části hry znamenají RFE šachty, kde se pohybují energetické bloky. Směr pohybu těchto bloků můžeme ovlivnit raketou.

Pokud zdárně zvládnete tuto část hry, objeví se vstup do druhého podzemního podlaží pevnosti Fort Apocalypse. Opět zde doplňte palivo. Pohyb chodbami se podobá první části pevnosti. Navíc se zde po vás požaduje, abyste pronikli ochranným štítem složeným z malých bloků energie, které se podobají cihlám, až k srdci pevnosti Fort Apocalypse - zdroji energie (nukleární reaktor). Po zničení reaktoru vyletí část pevnosti do povětří a vás po dobu návratu na zemský povrch neohrožují žádné tanky ani řízené střely.

Nezapomeňte po cestě sbírat ohrožené zajatce. Nevyhazujte reaktor do povětří, dokud nemáte všechny rukojmí na palubě vrtulníku. Počítač průběžně hlásí výsledky a oznamuje, kolik osob ještě musíte nalézt a zachránit. Využívejte také přistávacích plošin pro váš vrtulník - v případě zničení vrtulníku budete v další části hry pokračovat z této plošiny, nikoliv od začátku.

Hra Fort Apocalypse má také několik stupňů obtížnosti. Stiskem tlačítka OPTION se dostaneme do menu, kde si můžete vybrat kombinaci, která vám bude vyhovovat. Menu má tři možnosti volby :

1) úroveň gravitace ... slabá, normální, silná
   Gravity Skill ... weak ,normal , strong
2) úroveň pilota-hráče ... nováček, profesionál, expert
   Pilot Skill ... novice, pro, expert
3) počet vrtulníků, které máte k dispozici - 7, 9 nebo 11
   Robo Pilots - seven (7), nine (9), eleven (11)

Hra končí přistáním na téže plošině s palivem, odkud jste poprvé vzlétli. Počítač potom zkontroluje vaši činnost a ohodnotí ji patřičným stupněm podle kritérií US AIR FORCE.

Hra Fort Apocalypse vyžaduje koncentraci. Chce to silné nervy a zachovat klid i v takových situacích, kdy letíte s vrtulníkem klesajícím pod účinkem gravitace úzkou chodbou, zatímco jste atakováni různými nástrahami, které na vás nepřítel nachystal. Zručné hráče potěší toto napětí boje; ti méně obratní se dokonale zpotí.

Hra Fort Apocalypse je velmi dobře provedena a má citlivý ovládací a kontrolní systém. Grafika je velmi dobrá - využívá se zde možnosti předefinování znakové sady počítačů Atari. Jediným nedostatkem je, že mnoho hráčů ovládá tuto hru se slušným procentem úspěšnosti již za několik hodin. Přesto je hra Fort Apocalypse zábavná a dokáže si udržet váš zájem poměrně dlouhou dobu.

(c) 1988 by GIA Software
Jiří Hrdlička
Atari 130 XE

Literatura:

The Book of Atari Software 1984
(The Book Company, Los Angeles)

LETAJICÍ VRTULNIK

Program v BASICu demonstruje další možnosti PM-grafiky ve spojení s přerušením 50Hz. Krátká strojová rutina volaná při VBI provádí snímání joysticku pohyb vrtulníku a zvukové efekty.
Dílčí pohyby vrtulníku zajišťuje rutina periodickým přepisováním dvou podob vrtulníků, a to s pootočenými vrtulemi.

Popis programu:

Řádek 10 zajistí zvolení potřebného grafického režimu a nastavení barev. Řádky 30 až 90 nastaví parametry do systémových proměnných PM-grafiky (dvouřádkové rozlišení, zelenou barvu pro player 0 a 1, zápis adr. PMBASE). Řádky 100 až 110 vynulují paměť pro PMG. Řadeky 120 a 130 nastavují "systémové" proměnné (204,205 adresa tabulek poloh vrtulníku, 206 vertikální pozice vrtulníku, 207 adresa PMBASE). Řadek 130 nastavení horizontální pozice. Řádeky 200 až 210 načtení dat strojového kódu. Řádek 215 nastavení ukazovátka VBI na rutinu. Řádek 216 nastavení zvukového generátoru. Řádky 219 až 500 data strojového kódu. Řádky 1000 až 2600 data poloh vrtulníku.

Petr Šťastný
AK Olomouc

Listing prog.VRTULNIK.BAS

```
XP 10 GRAPHICS 0:SETCOLOR 2,14,0:SETCOLOR
   1,0,7
LR 30 POKE 559,62
PO 50 POKE 704,188:POKE 705,188
FA 60 POKE 623,1
DM 70 I=PEEK(106)-16
BZ 80 POKE 54279,I
NY 90 POKE 53277,3
IN 100 J=I*256+1024
PL 110 FOR I=J TO J+512:POKE I,0:NEXT I
HH 120 POKE 204,0:POKE 205,100:POKE 206,5
   0:POKE 207,J/256:POKE 1788,5:POKE 1790,0
II 130 POKE 53248,120:POKE 53249,128:POKE
   1791,120
IP 200 RESTORE 200:FOR I=0 TO 1000:READ D
   :IF D=-1 THEN 215
```

```
NX 210 POKE I+1536,D:NEXT I
CB 215 U=USR(ADR("h" )L\d")):REM POKE
   548,0:POKE549,6
CC 216 SOUND 3,0,0,7
FD 219 DATA 173,0,211,74,176,13,166,206,2
   24,20,240,7,160,3,140,252,6,198,206,74
   ,176,8,166,206,224,220,240,2,230,206
KN 220 DATA 206,253,6,208,43,173,252,6,14
   1,253,6,173,251,6,208,17,24,165,204,10
   5,128,133,204,169,4,141,251,6,141,6
RT 230 DATA 210,208,15,56,165,204,233,128
   ,133,204,169,0,141,251,6,141,6,210
PF 240 DATA 173,0,211,74,74,74,176,46,174
   ,255,6,224,40,240,39,172,254,6,192,1,2
   40,4,192,9,208,7,165,204
EK 250 DATA 56,233,32,133,204,192,0,240,3
   ,206,254,6,202,142,255,6,138,24,141,0,
   208,105,8,141,1,208
EO 260 DATA 173,0,211,74,74,74,74,176,46,
   174,255,6,224,220,240,39,172,254,6,192
   ,8,240,4,192,0,208,7,165,204
AN 270 DATA 24,105,32,133,204,192,9,240,3
   ,238,254,6,232,142,255,6,138,24,141,0,
   208,105,8,141,1,208
HW 480 DATA 56,165,206,233,16,133,206,230
   ,207,160,31,177,204,145,206,192,16,208
   ,9,198,207,24,165,206,105,16,133,206
OB 490 DATA 136,16,236
OE 499 DATA 169,5,141,252,6
MX 500 DATA 76,98,228,-1
TS 1000 RESTORE 1000:FOR A=25600 TO 25600
   +95:READ D:POKE A,D:NEXT A
FO 1010 DATA 0,255,1,1,15,17,33,49,127,11
   2,63,31,16,160,127,0
PO 1020 DATA 0,192,0,1,1,129,255,14,122,9
   8,194,192,64,32,252,0
HF 1030 DATA 0,255,1,1,3,7,12,8,16,27,15,
   7,8,16,56,0
QL 1040 DATA 0,192,0,0,128,192,96,32,16,1
   76,224,192,32,16,56,0
GV 1050 DATA 0,3,0,128,128,129,255,112,94
   ,70,67,3,2,4,63,0
ZI 1060 DATA 0,255,128,128,240,136,132,14
   0,254,14,252,248,8,5,254,0
WI 2000 RESTORE 2000:FOR A=25728 TO 25728
   +95:READ D:POKE A,D:NEXT A
VX 2010 DATA 0,7,1,1,15,17,33,49,127,112,
   63,31,16,160,127,0
EM 2020 DATA 0,254,0,2,2,130,254,15,121,9
   7,193,192,64,32,252,0
FK 2030 DATA 0,7,1,1,3,7,12,8,16,27,15,7,
   8,16,56,0
```

Potřebujete 16-bitů?

Matthew J. W. Rattcliff

Dnes je jasné, že každý chce počítač Atari ST. A proč ne? Má 512 barev, grafiku s vysokou rozlišovací schopností a nový výkonný 16-bitový mikroprocesor. Je tedy vaše 8-bitové Atari XL nebo XE zastaralé? Rozhodně ne! Pokud váš 8-bitový počítač zvládne dokonale všechny vaše požadavky, nepotřebujete přece kupovat mnohem výkonnější 16-bitový počítač. Musíte se proto sám sebe zeptat: chci ST?; mohu si dovolit ST?; a potřebuji opravdu ST?

S prvními dvěma otázkami si musíte poradit sami. Doufám však, že vám pomohu s třetí otázkou. A ty z vás, kteří máte strach ze zastarávání vašeho 8-bitového systému, ubezpečím, že tato obava není na místě.

Větší je lepší!? Američané (a nejen oni) mají tendenci myslet a jednat tímto způsobem, který není vždy nejlepší. Například, čím výkonnější je mikroprocesor, tím více je komplikovanější. Profesionální software 16-bitových počítačů dokáže zpracovat obrovské množství informací za nepatrnou dobu, ale také trvá delší dobu program napsat, odstranit chyby (ty se najdou vždycky) a vyladit program do optimální podoby. Můžete se tedy vsadit, že software bude mnohem dražší a méně přístupnější než na 8-bitové počítače. A to, naneštěstí, nemusí být vždy výsledek dokonalý a mít ceně odpovídající výkon. Protože 16-bitový procesor může adresovat mnohem více paměti, programátoři mají tendenci přiklánět se k tomuto postoji: "Proč optimalizovat zdrojový kód? Máme přece velkou paměť." Konečným produktem bývá pomalý program, který "pustoší" a zabírá velkou část paměti. Také většinou nevyužije přednosti a výkonnost mikroprocesoru. To na 8-bitových počítačích musí být naprostá většina dobrých programů optimalizována kvůli menší kapacitě paměti a účinnosti procesoru. Taková "neprodyšná" paměť dokonce programy také zrychluje!

ST Basic má plno zbytečných ozdob. Plně sice podporuje grafické možnosti systému a okénkovou metodu zpracovávání až k pocitu marnivosti. Zato logicky spojené funkce, jako je editace a výpis programu, se nachází v jiném okénku - divné pojetí! Je to také pomalý Basic. Každý, kdo pracoval nebo viděl práci s ST Basicem, byl zklamán. Nepracuje o mnoho rychleji než Basic na 8-bitových počítačích. Proč? A opět jsme u toho, větší není vždy lepší. ST Basic podporuje mnoho nových příkazů a instrukcí, což znamená, že mikroprocesor musí dlouho hledat "překlady", než spustí váš program.

Vytvoření softwaru na ST je obtížnější než na 8-bitové počítače. Pravda, jazyk C nebo assembler 68 000 pro Atari ST jsou mnohem výkonnější dialekty jazyků než všechno, co můžete najít pro procesor 6502. Ale ta odlišnost v kompilaci programů! Všechno na ST je umístěno na jednotlivých discích, zvlášť editor, kompilátor, slučovač programů a podpůrné programy pro odstraňování chyb. Stále nahráváte programy a vytváříte nové

soubory, dokud nedostanete konečný kód. A když hotový program nefunguje? Zpátky do editoru a začít celý proces znovu od začátku.

My, 8-bitoví programátoři, se máme zatím lépe. Firma OSS pro nás zhotovila program MAC/65, což je "de fakto" standart v rozvoji softwaru pro 8-bitové Atari. Je to také nejrychlejší dosud vyrobený kompilátor pro 6502. Tento assembler přeloží 40 KB zdrojového kódu do výkonného strojového jazyka za méně než 10 sekund! Program MAC/65 zahrnuje editor, assembler a debugger (odstraňování chyb) v jednom. Každý modul je přístupný na povel.

Samozřejmě, nový software pro ST se stále ještě vytváří. Určitě brzy budou na trhu rychlejší a přizpůsobivější podpůrné programy a kompilátory pro Atari ST (například Basic GFA je opravdu extrémně rychlý). Jen si vzpomeňte, 8-bitové počítače Atari byly několik let "mimo", než se programy MAC/65 a Basic XL/XE staly hitem na počítačovém trhu.

Jestliže jste nespokojeni s procesorem 6502 a nedostatkem kapacity paměti pro "vážné" aplikace programového vybavení, koupě počítače Atari ST je pro vás dalším logickým krokem. Ale pozor - přechodem na procesor 68 000, po několikaleté práci s procesorem 6502, je jako přechod z VW Brouka na Porsche 944 Turbo - zůstávají pouze "základní řidičské zkušenosti".

Touto novou konkurencí klesá na ceně software pro 8-bitové počítače Atari. Obáváte se, že nebude dostatek nového softwaru pro váš 8-bitový počítač? Zbytečné obavy. Potřebujete lepší textový procesor než je Paperclip nebo Atari Writer Plus? Ne! Více paměti? Dostali jste někdy na obrazovku zprávu, že nemáte více paměti pro váš Basicový program? Já ne, i když jsem delší dobu vlastnil jen Atari 400 (16 KB RAM). Pokud však potřebujete více místa na programování, jazyk Basic XE spolu s počítačem Atari 130 XE vám dá přes 60 KB volné paměti a 30 KB pro uskladnění proměnných a polí. To vyjde jistě levněji, než koupit systém ST.

Potřebujete lepší grafiku? Jestliže ano, možná jste nikdy nehráli hry firmy Epyx (Rescue on Fractalus, Koronis Rift nebo Eidolon). Pokud je však lepší grafika podmínkou, jistě rádi uslyšíte, že počítač Atari 520 ST má vyšší rozlišovací schopnost v monochromním modu než Apple Macintosh a že kompletní barevný systém 520 ST plus rozšíření na 1 MEG RAM je levnější než rozšíření paměti na 1 MEG RAM pro Apple Macintosh.

Potřebujete výkonnější databázi, rychlejší soubory vstupu a výstupu a možnost připojení více standartních interfaců? Potom počítače řady ST splní všechna vaše přání. Potřebujete interface MIDI pro váš syntezátor CASIO CZ 101? Atari ST ho má zabudovaný. Pokud chcete předvádět kouzla s assemblerem, procesor 68 000 v Atari ST je asi 20 krát výkonnější než mikroprocesor 6502 - a pracuje na 8 MHz, tedy 8 krát rychleji než 8-bitové Atari.

Potřebuji ST? Možná ne - ačkoliv, za tu cenu (bohužel jen v západních státech, nikoliv u nás - pozn. překladatele), nemůžete nakoupit více hrubé počítačové síly.

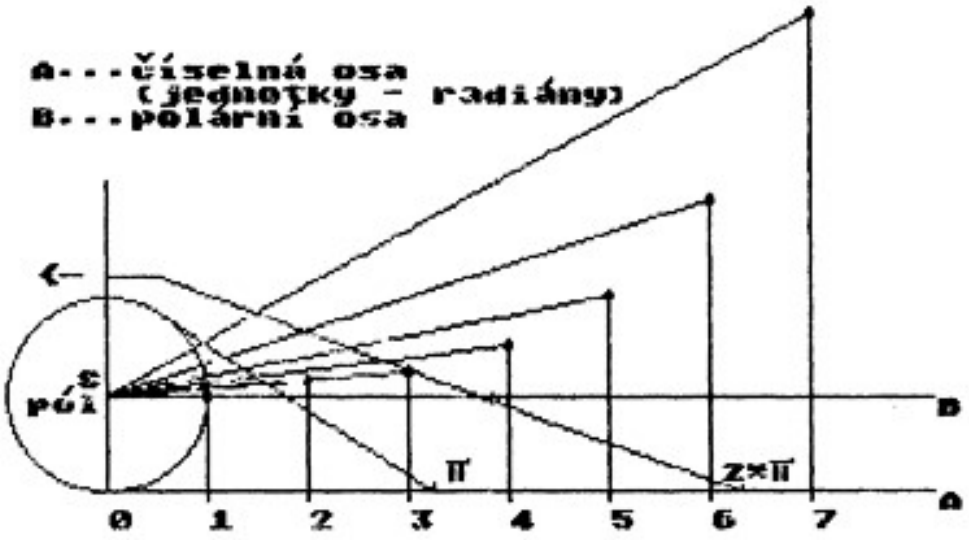
A co já? Samozřejmě, že mám Atari ST. Ne však proto, že by počítač Atari 130 XE nezvládl to, co potřebuji. Ale jako

"advokát" firmy Atari v počítačovém světě musím jít s pokrokem. Stále však budu podporovat programování v assembleru na počítači Atari 130 XE s programem MAC/65. A dobrá rada na závěr - pokud přejdete na počítač ST a dáte 8-bitové Atari "do výslužby" nebo ho prodáte, budete brzy tuze litovat...

(c) 1988 by GIA Software
Jiří Hrdlička
Atari 130 XE

Program Polar Plotter II

Vysvětlující obrazek k textu článku:



Atari XL/XE - náhradní zdroj na baterie

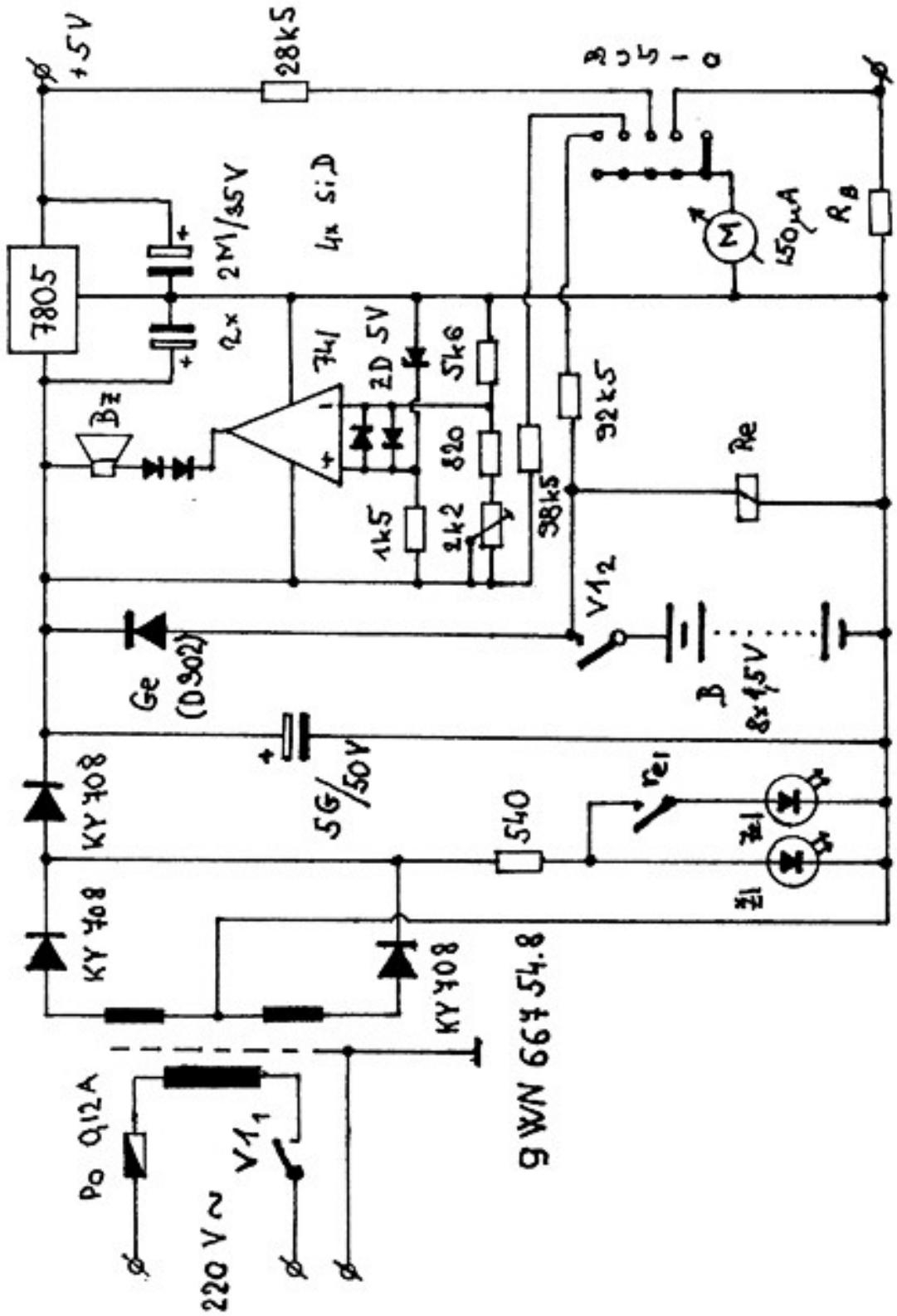
ing. Jiří Hrdlička

Při programování je velmi nepříjemný výpadek síťového napájení. Prakticky to znamená začít dělat program znovu. Majitelé disketových jednotek to mají jednoduché - programy dělí na části a ty potom průběžně zaznamenávají na disk. Pro uživatele magnetofonů je však tato manipulace složitá a při delším programu i časově náročná.

Přitom 8-bitové počítače Atari jsou napájeny jen jedním napětím 5 V o celkem malé intenzitě asi 0,8 A. Dá se tedy i s použitím našich monočlánků vyrobit bateriově zálohovaný zdroj k počítači.

Při přítomnosti sítě je počítač napájen přes síťové trafo, napětí je usměrněno, vyhlazeno a pomocí integrovaného stabilizátoru MAA 7805 stabilizováno na 5 V. Usměrněné síťové napětí na vstupu stabilizátoru musí být větší než napětí baterie. Při výpadku síťového napětí převezme napájení (bez přepínacích výpadků napětí i špiček napětí) baterie, složená z osmi monočlánků 1,5 V (Pro dobrou funkci stabilizátoru je zapotřebí alespoň 6,6 V a pro dobré využití kapacity baterie je uvažován pokles napětí jednoho článku na 0,82 V.). Přítomnost síťového napětí je signalizována zelenou LED diodou, napájení z baterie je signalizováno žlutou LED diodou přes relé (kvůli odběru proudu - polarizované relé odebírá 0,6 mA). Zdroj je doplněn měřícím přístrojem, který měří odběr proudu, napětí 5 V, napětí baterie a napětí na vstupu stabilizátoru MAA 7805. Dále je doplněn signalizačním obvodem, který dává akustický signál při poklesu napětí na vstupu stabilizátoru na 6,7 V (při tomto poklesu je nutné nahrát to, co máme v paměti počítače na msf. kazetu - za cca 5 minut napájecí napětí poklesne pod 5 V).

Podle stavu napájecích baterií je možné počítat s provozem 0,5 - 1 hodinu. Tento zdroj by měl být dostupný v klubech - jednotlivec jej dostatečně nevyužije a je pro něj zbytečně drahý. Jinak splňuje všechny na něj kladené požadavky. Použité součástky jsou popsány ve schématu, zdroj byl postaven v olomouckém Atari klubu.


+5V
28k5
7805
2M1/35V
2x
4x SiD
741
ZD 5V
5k6
820
1k5
2k2
38k5
92k5
Bz
150μA
Re
V1₂
Ge (D902)
8x1,5V
5G/50V
KY 708
540
rel
žl
Po 012A
220 V~
V1₁
9 W/N 667 54.8

MURPHOLOGIE

ZÁKLADNÍ ZÁKON MURPHYHO

Co se může pokazit, to se taky pokazí.

Důsledky:

1. Nic není tak jednoduché, jak se to na první pohled zdá.
2. Všechno stojí více času, než jsi předpokládal.
3. Může-li se něco pokazit ze čtyř různých příčin a podaří-li se ti nakrásně všechny čtyři odstranit, vzápětí se objeví pátá.
4. Když se do něčeho dáš, najde se něco, co musíš udělat ještě dřív.
5. Příroda nadržuje skrytým vadám.
6. Matička příroda je děvka.

SCHNATTERLYHO SHRNUTÍ DŮSLEDKŮ

Co se nemůže pokazit, to se taky pokazí.

REVIZE KVANTOVÉHO ASPEKTU MURPHYHO ZÁKONA

Když se něco pokazí, pokazí se všechno naráz.

SCHNATTERLYHO SOUHRN LOGICKÝCH DŮSLEDKŮ

Jestliže něco nemůže dopadnout špatně, stejně to dopadne špatně.

ROZŠÍŘENÝ MURPHYHO ZÁKON

Řetězec nešťastných událostí se zpravidla uspořádá v co možná nejméně příznivém sledu.

O TOOLOVA POZNÁMKA K MURPHYHO ZÁKONU

Murphy byl optimista.

ZAKONY PROGRAMOVANÍ

1. Každý program je zastaralý ve chvíli, kdy je vložen do počítače.
2. Příprava každého programu stojí vždy víc a trvá vždy déle.
3. Jestliže se program osvědčí, je třeba jej změnit.
4. Jestliže se program neosvědčí, je třeba o něm pořídit dokumentaci.

GRAYOV ZAKON PROGRAMOVANÍ

Vykonání n+1 prkotin vyžaduje přesně tolik času jako vykonání n prkotin.

LOGGOVO VYVRACENÍ GRAYOVA ZAKONA

Na vykonání n+1 prkotin se spotřebuje dvojnásobné množství času než na vykonání n prkotin.

ZAKLADNÍ PRAVIDLO PRO PROGRAMATORY

Když už nerozumíš tomu, co děláš, dělej to alespoň precizně.

CAHNOV AXIOM

Selhaly-li všechny pokusy, je načase přečíst si návod.

GREEROV TRETÍ ZAKON

Počítačový program dělá jen to, co mu řeknete, nikdy však nedělá to, co byste chtěli, aby udělal.

SUTTINOV DRUHÝ ZAKON

Největší zábavu skýtají ty počítačové úlohy, které nejsou v praxi naprosto k ničemu.

MURPHYHO ČTVRTÝ LOGICKÝ ZAVER

Udělat cokoli blbovzdorným je nemožné, protože blbci jsou ohromně vynalézaví.

PATÝ ZAKON SPOLEHLIVOSTI

Mýlit se je lidské. Ale něco dokonale zašmodrchat je možné pouze s pomocí počítače.

ZAKON H. L. MENCKENA

Kdo to umí, ten to dělá. Kdo to neumí, ten to učí.

MARTINOV DOPLNEK

Kdo to neumí učit, ten to řídí.

MURPHY S LAW DESK CALENDAR / Los Angeles, USA
(c) 1988 by GIA Software
Jiří Hrdlička
Atari 130 XE

Obsah Atari zpravodaje Olomouc 1-2/1989:

Redakční oznámení:

!!! PROGRAMATORI - POZOR !!!

Hledáme autory článků - SOFTWARE i HARDWARE 8-bitových počítačů Atari XL/XE. Neváhejte a pošlete článek do naší redakce. Přivítáme i překlady ze zahraniční literatury. Vždy však uvádějte, na základě jaké publikace nebo časopisu jste článek zpracovali. Honorář máme možnost vyplácet v rámci směrnice pro hospodaření ZO Svazarmu.

Od roku 1989 začínáme zpracovávat veškeré informace, které se objeví v našem zpravodaji, na počítačích Atari. Prosíme Vás tedy o následující:

Příspěvky zasílejte pokud možno napsané v programu Čapek (verze 1.1 nebo 2.x) a uložené na magnetofonové kazetě (nejlépe C-60) nebo disku. Raději svůj příspěvek uložte 2x nebo 3x za sebou, totéž platí i v případě programu. Během tří týdnů se Vám kazeta nebo disk vrátí. Značně tím urychlíte průběh zpracování jednotlivých čísel Zpravodaje a pomůžete zlepšit namáhavou práci redakční rady.

Zásilky posílejte na adresu olomouckého Atari klubu a označte je nápisem ZPRAVODAJ.

Naše adresa: Atari klub Olomouc
PS 137
772 13 OLOMOUC

Zpravodaj AK Olomouc 1/2 1989
NEPRODEJNÉ, odběr vázán na příspěvek.

Odpovědný redaktor: Ing Kopečný Pavel
Odborný redaktor : Hrdlička Jiří

Neprošlo jazykovou úpravou.
Předáno do tisku: I.-1989

Tisk povolen OK ONV Olomouc, 0380500387
Tisk: MTZ Olomouc, z 50

30-;