SoftSide™

May 1982 Vol. 5, No. 8 THREE DOLLARS

Freefall Inc.

100+ Reasons

DOS FEATURES

- A 252+ page technical manual with index and detailed table of contents.
- Commands **SYSTEM** and **PDRIVE** allow the user to configure/customize his/her own DOS
- Depending on installed hardware, **NEWDOS/80,** via the **PDRIVE** command, supports within one system, mixtures of single/double density, single/double head, 5 or 8 inch drives with single/double volume diskettes of up to 7680 standard 256 byte sectors. 80 track drives can read 35/40 track diskettes. Parameters for 10 drives may be prespecified though only a maximum of 4 are in use at anytime. ADR not provided.
- LNW 5/8 and Omikron mapper boards supported.
- APPARAT, AEROCOMP, AM, LNW, and PERCOM, disk doublers supported.
- Most CPU speed up mods may be used though not specifically supported.
- Model I/III data diskette interchangeability when both computers are operating under **NEWDOS/80** version 2.0.
- Model I 2.3 TRSDOS compatible.
- **COPY** to/from Model I 2.3B and Model III TRSDOS diskettes (no other useage allowed).
- Single drive **COPY** and Copy By File features.
- Depending on installed hardware, the system volume may be single/double density, single/double sided, 5 or 8 inch and up to 7680 sectors.
- **RUN-ONLY** program mode restricts the operator to program defined input only.
- **MINI-DOS** allows the executing program to be interrupted by the operator to perform one or more of the 51 DOS commands executable under MINI-DOS, and then continue the interrupted program's execution.
- **CHAIN** or **DO** commands activate chaining whereby keyboard input comes from the specified disk file, allowing a pre-determined set of commands and/or parameters to be automatically inputted.
- Dump display to printer function.
- Enhanced **DEBUG** facility (14 commands) allows interrupting current program execution, inspecting/altering memory or disk, and resuming execution, continuous or single step, with/without stops.
- DOS vectors defined for Assembly Language programmers.
- DOS-CALL allows user programs and BASIC to execute DOS commands.
- The programmer may create his own resident DOS commands.
- Programs may enable/disable user routines driven off the timer interrupt.
- The programmer may create his/her own resident DOS commands.
- Model I built-in lower case driver, blinking cursor, auto key repeat.
- **ROUTE**ing of keyboard, display, printer and (Model III only) RS232C. May be routed to a user routine in memory, but not to/from disk unless via a user routine.
- Except for the spooler, there are no high memory routines for DOS or BASIC; this includes **ROUTE** and **CHAIN** functions.
- Lower case DOS commands honored.
- Full error messages displayed instead of error codes.
- 31 enhanced **COPY** parameters.
- Copy By File allows 6 criteria for file selection.
- 15 enhanced **FORMAT** parameters.
- Partial diskette re**FORMAT** permitted.
- File **PURGE** by wildcard extents and/or user files.
- **DI**Rectory command allows wildcard extents, user files, short or extended format, dump to printer.
- User may specify diskette's directory location.
- Expanded directory provides for up to 222 file entries.
- Some DOS commands may be aborted without reset.
- **R** command repeats last performed DOS command.
- **CREATE** command to pre-allocate a disk file.
- **ERROR** command displays error message associated with error code.
- **HIMEM** command sets/displays DOS/BASIC high memory address.
- **DATE** command sets/displays computer's date.
- **TIME** command sets/displays computer's time.
- Model III **FORMS** command for printer control.
- Model III **SETCOM** command for RS232 control.
- Enhanced **LIST/PRINT** commands for ASCII files with pause, abort and partial file listing.
- Alter chaining state via the **CHNON** command or commands within the chain file.
- A program or a chaining sequence may display a message with/without pause.
- **CLEAR** command to zero memory and to purge routes, user DOS commands and user timer routines.
- Commands to enable/disable BREAK key, blinking cursor, lower case driver
- **PROT** command to change diskette **NAME/DATE/PASSWORD**
- **ATTRIB** command to change a file's attributes.
- **CLS** command to clear screen.
- **AUTO** specifies the command to execute automatically at reset/power-on.
- **SYSTEM** specifies the default system configuration values (usually enable or disable) which become effective on RESET/POWER UP.

  Diskette/file password checking
  RUN-ONLY mode
  Keyboard debounce (Model I)
  Screen dump to printer (JKL)
  DEBUG 123 entry
  MINI-DOS
  Break key as keyboard key
  Hardware lower case (Model I)
  Assign default drive number for **DIR**
  Assign default drive number for file creation
  Memory protect value
  Clear key as keyboard key
  Disk master password required for full diskette or **CBF COPY**
  Auto Repeat key function
  TIME/DATE question on power-up
  TIME/DATE question on reset
  Display disabled until operator/program reenables
  Manual operator chaining pause/abort
  Manual operator **AUTO** command override
  R = repeat last DOS command performed
  Built-in lower case driver (Model I)
  Lower to upper case toggle
  Blinking cursor
  Number of physical drives on computer
  Number of disk I/0 retries
  Time delay for 1st repeat of auto repeat key
  Specify the cursor character
  Specify the write of the directory sector's address mark for Model I single density diskette in Model III format for easy Model I, Model III diskette exchangeability

Apparat, Inc.

800/525-7674 (order line only)
303/741-1778
4401 South Tamarac Parkway
Denver, Colorado 80237

Why NEWDOS/80™ VER. 2.0 Is the Best DOS for your TRS-80 Model I or III Computer

DISK BASIC FEATURES

- In one statement from DOS READY, BASIC can be brought up, the number of files set, the memory size set and a program LOADed or RUN.
- **RUN-ONLY** prevents the operator from getting to READY or DOS READY, thus giving the program almost total control.
- Via the CMD function, all DOS commands are executable from BASIC, either directly or dynamically.
- **MINI-DOS** is available from BASIC.
- **DEBUG** is available from BASIC.
- **CHAIN**ing is available from BASIC.
- Variable passing between programs via the **V** parameter of RUN.
- Abbreviated commands:
  (A)uto; **(D)**elete; **(E)**dit or comma; **(L)**ist or period
- Accidental text line deletion more difficult
- Text line scrolling foreward or backward.
- Text page scrolling foreward or backward.
- **DI** moves text line to new position
- **DU** duplicates text line to new position
- Built-in **RENUM**ber with line number and limited syntax error check. A portion of text may be moved to another part of the program with all references to that code resolved.
- Built-in **REF**erence function will display/print references to all line numbers, integers and variables. It will display references to a single line number, integer, string, function code (reserved word) or a group of packed or unpacked characters, and then allows displaying of each referencing text line in turn with editing as necessary.
- A program may be loaded into reserved high memory via CMD or MINI-DOS and its execution address extracted from the two bytes at 17411 (4403H).
- **MERGE** functions with packed or ASCII text files.
- Built-in text space eliminator and/or remark deleter.
- Built-in calendar date conversion.
- Dynamic **ERASE** of selected variables, keeping all others.
- Dynamic **KEEP** selected variables, **CLEAR**ing all others.
- After clearing an array via **ERASE** or **KEEP**, the array may be redimensioned via **DIM.**
- Dynamic text line deletion.
- Dynamic text line insertion via **MERGE,** which with dynamic **DELETE,** allows use of overlays.
- **SWAP** contents of 2 variables of the same type.
- Single Stepping starting at specified text line number.
- In-memory sort of up to 9 arrays in either ascending or descending order
- **RENEW** function to reinstate **NEW**ed program.
- Full BASIC error messages, including associate DOS error message, if applicable
- With default start up parameters and no reserved high memory, 48K RAM has 38261 bytes available.
- SUPERZAP, DIRCHECK and other programs using only memory from 5200H to 6FFFH can be executed directly from BASIC without disturbing the program text or variables (if 8K BASIC free memory available, exclusive of string area).

FEATURES of NEWDOS/80 enchanced BASIC disk file I/O.

- In addition to TRSDOS sequential and random file types, **NEWDOS/80** has two new file types (Marked Item and Fixed Item) divided into five subtypes **(MF, MU, MI, FF** and **FI)**
- These five subtypes do not require **LSET, RSET, MKI$, MKS$, MKD$, CVI, CVS** or **CVD**; instead, **GET**s and **PUT**s are done directly to/from the variables named in a list.
- The string separating character sequence ; " , " ; used with PRINT is not used with the new file types; instead only a comma is used as the separator.
- **MU** files are used as an option to the older PRINT/INPUT files.
- **FF** files are used as an option to the older RANDOM files.
- Record lengths up to 4095 bytes supported.
- Records may be all of the same length (**MF** and **FF**), of varying lengths (**MU**) or unknown length (**MI** and **FI**).
- Sequential files may be accessed randomly.
- Files may be accessed by **R**elative **B**yte **A**ddress to allow accessing of variable length or unkown length records.
- Existing files may be extended.

UTILITY PROGRAMS INCLUDED WITH NEWDOS/80

- **SUPERZAP** is a disk/memory display and modification program, also used as the vehicle for installing patches to **NEWDOS/80.**

- **DISASSEM** is a Z80 load module (CMD) disassembler that builds cross reference tables for all location references including those by JR instructions, includes in the disassembly printable characters for all hex bytes to help locate character strings and sends the disassembly to the display, printer or a disk file. The disk file can then be edited and/or assembled using EDTASM, if it is not too large.

- **DIRCHECK** is a program that displays directory contents and checks directory integrity (its primary function), displaying specific error codes to assist user attempts at directory trouble shooting and/or repair. Optionally will zero out unused (dead) file names.

- **EDTASM** is Apparat's enchancement of Radio Shack's 1978 tape editor/assembler program to operate from disk and with disk files. Requires purchase of that Radio Shack program (not a newer one) as a pre-condition of use of Apparat's EDTASM.

- **LMOFFSET** allows load module (CMD) transfer between disk and tape. Displays program start, end and entry addresses. Optionally allows load address relocation (not execution relocation) and subsequent execution as from non-disk BASIC via SYSTEM.

- **CHAINBLD** is a mini-text editor for creating/maintaining chaining files.

- **NEWDOS/80** manual chapter titles and page counts
  1. Introduction (5)
  2. DOS Library Commands (52)
  3. DOS Routines (12)
  4. DOS Features (14)
  5. DOS Modules, Data Structures, and Miscellaneous Information (12)
  6. Additional Programs Supplied on **NEWDOS/80** Diskette (22)
  7. Disk BASIC, non-I/O Enhancements (17)
  8. Disk BASIC I/O Enhancements and Differences (21)
  9. Error Codes and Messages (2)
  10. Glossary (9)
  11. Error Reporting, Incompatibility Handling, and Patching (8)
  12. Conversion Information and Miscellaneous Comments (9)
  13. ZAPs (increasing with time)
  14. Appendix A: Marked and Fixed Item File discussion (47)
  15. Appendix B: Marked and Fixed Item File examples (18)
  16. Index (4)

- Full time support staff
- Information, minor enhancements and corrections to **NEWDOS/80** are issued, at no charge, to registered owners only.

NEWDOS/80 Version 2.0 for the model I is a separate and distinct product from the model III. Each is sold separately.

$149.00

"On-going Support for Microcomputers"

®REGISTERED TRADEMARKS OF APPARAT, INC.

SoftSide™

PUBLISHER
Roger W. Robitaille, Sr.

ASSOCIATE PUBLISHER/EDITOR
Randal L. Kottwitz

MANAGING EDITOR
ART DIRECTOR
Nancy Lapointe

SOFTWARE MANAGER
Bill Kubeck

PROGRAMMING EDITOR
Jon Voskuil

EDITORIAL DEPARTMENT
Rich Bouchard
Lance Micklus
Mark Pelczarski
Joan Truckenbrod
Alan J. Zett

ADVERTISING MANAGER
Sue Rowland

ART PRODUCTION MANAGER
Lynn Wood

PRODUCTION DEPARTMENT
Lynda Fedas
Denise Lafleur

CUSTOMER SERVICE
Cindy Schalk

DEALER SALES
Kathie Maloof

STAFF
Mary Edwards
Donna Jean
Bea Kimball
Doris Miller
Cindy Schalk

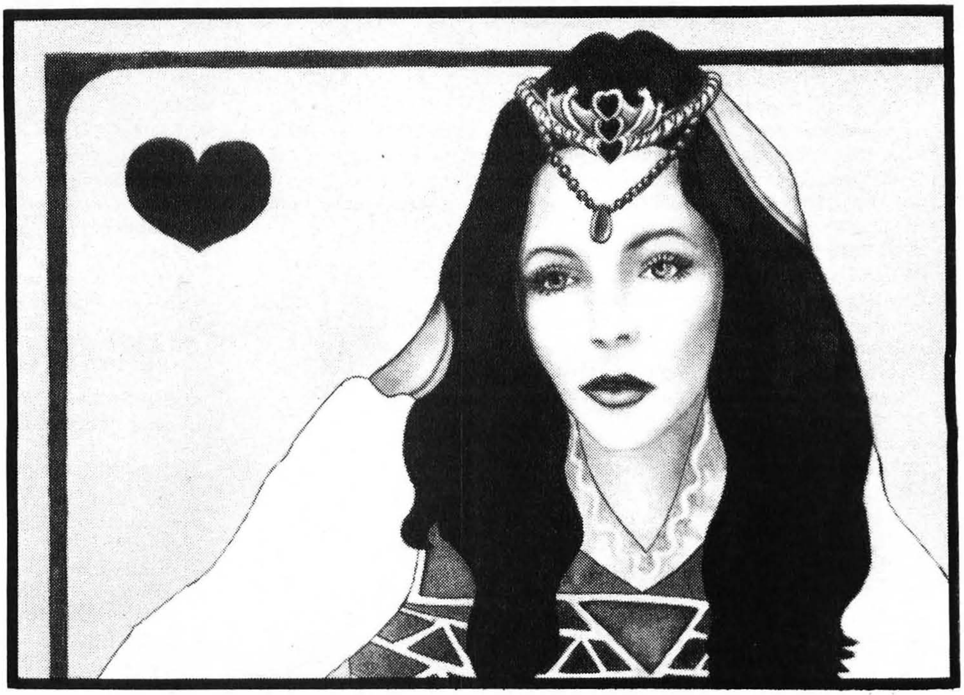
Photographs by Dean F. H. Macy

SoftSide is published each month by *SoftSide* Publications, 6 South Street, Milford, New Hampshire 03055. Telephone 603-673-0585. Second class postage paid Milford, New Hampshire and pending at additional mailing offices. ISSN: 0274-8630. Subscription rates: USA and Canada, $30.00/12 issues. First Class USA, APO, FPO, Mexico, $40.00/12 issues. Other foreign countries, $62.00/12 issues. Media subscription rates: For USA magazine and cassette, $75.00/12 months. Magazine and disk, $125.00/12 months. APO, FPO, Canada, Mexico, add $20.00/12 months. Other foreign add $50.00/12 months. All remittances must be in U.S. funds. Mail subscription inquiries to *SoftSide* Publications, 515 Abbot Dr., Broomall, PA 19008

POSTMASTER - send address changes to:

SoftSide Publications
515 Abbot Drive
Broomall, PA 19008

If you have not received your June issue of *SoftSide* by June 6, contact *SoftSide* Publications, 515 Abbot Drive, Broomall, PA, 19008 or call 1-800-345-8112 (In PA call 1-800-662-2444).



FRONT RUNNER

FEATURES

DEPARTMENTS

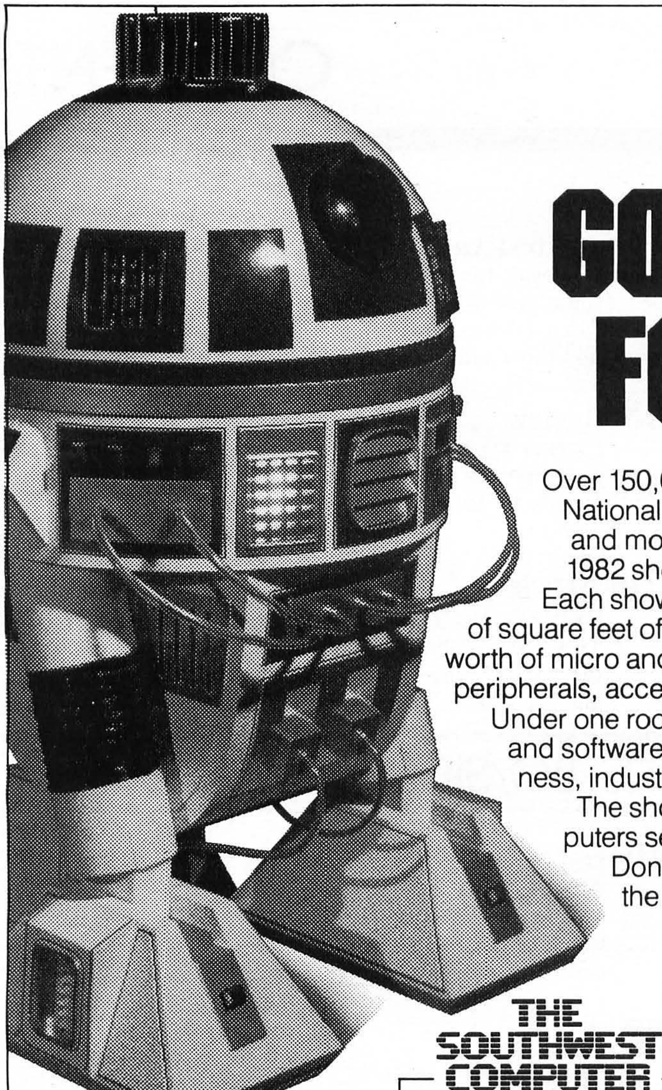
CONTENTS

APPLE/SIDE

ATARI/SIDE

0 — 19

TRS-80/SIDE



HAVE WE GOT A PROGRAM FOR YOU IN '82

Over 150,000 computer owners and novices attended the 1981 National Computer Shows and Office Equipment Expositions, and more than a quarter of a million are expected to be at the 1982 shows.

Each show features **hundreds** of companies using **thousands** of square feet of display space to showcase and sell **millions** of dollars worth of micro and mini computers, data and word processing equipment, peripherals, accessories, supplies and software.

Under one roof you'll see — and be able to buy — all of the hardware and software made by every major computer manufacturer for business, industry, government, education, home and personal use.

The show includes computers costing as little as $100 to computers selling for $150,000.

Don't miss the coming of the new computers — show up for the show. Admission is $5 per person and $3 for children.

THE NATIONAL COMPUTER SHOWS

Ticket Information

Send $5 with the name of the show you plan to attend to National Computer Shows, 824 Boylston Street, Chestnut Hill, Mass. 02167. Tickets can also be purchased at the show.

THE SOUTHWEST COMPUTER SHOW

Dallas
Dallas Market Hall

Thursday-Sunday
April 15-18, 1982
11 AM to 6 PM Daily

DIRECTIONS:
2200 STEMMONS FREEWAY
(AT INDUSTRIAL BLVD)

THE NEW YORK COMPUTER SHOW

Uniondale, Long Island
Nassau Coliseum

Thursday-Sunday
April 22-25, 1982
11 AM to 6 PM Daily

DIRECTIONS: TAKE L.I. EXPWY TO EXIT 38 NO. STATE PKWY TO EXIT 31A MEADOWBROOK PKWY SO. TO EXIT M5 HEMPSTEAD TURNPIKE

THE TWIN CITIES COMPUTER SHOW

Minneapolis
Minn. Auditorium
& Convention Hall
Third Avenue

Thursday-Sunday
September 16-19, 1982
11 AM to 6 PM Daily

DIRECTIONS: HWY 94 to *11th St. Exit to Third Ave.*

THE MID-ATLANTIC COMPUTER SHOW

Washington, DC
DC Armory/Starplex
Across from RFK Stadium

Thursday-Sunday
October 28-31, 1982
11 AM to 6 PM Daily

DIRECTIONS:
2001 E. CAPITOL ST. SE
(E. CAPITOL ST. EXIT OFF I-295 — KENILWORTH FRWY)

THE MID-WEST COMPUTER SHOW

Chicago
(Arlington Heights)
Arlington Park Racetrack
Exhibition Center

Thursday-Sunday
November 5-7, 1982
11 AM to 6 PM Daily

DIRECTIONS: EUCLID AVE & WILKE RD. TAKE NW TOLLWAY TO RTE 53 EXIT AT EUCLID AVE EAST

THE NORTHEAST COMPUTER SHOW

Boston
Hynes Auditorium/
Prudential Center

Thursday-Sunday
November 11-14, 1982
11 AM to 6 PM Daily

DIRECTIONS: TAKE MASS PIKE TO PRUDENTIAL CENTER EXIT

THE SOUTHEAST COMPUTER SHOW

Atlanta
Atlanta Civic Center

Thursday-Sunday
December 9-12, 1982
11 AM to 6 PM Daily

DIRECTIONS:
395 PIEDMONT AVE NE
(AT RALPH McGILL BLVD)

The National Computer Shows are produced by Northeast Expositions Inc. who also produce Electronica — shows featuring home entertainment equipment and personal electronics — which are held annually in major US cities. NEI also produces the Applefest Shows. For more information about any of these events call us at 617-739-2000 or write to the above address.

EDITORIAL

By Jon R. Voskuil

One thing I have noticed lately. People have an awful lot of opinions about an awful lot of things. This sometimes puts me in an awkward position when I sit down to write editorials, because I'm basically wishy-washy. And what good is an editorial if it doesn't rant and rave about something or another?

The solution to this problem was suggested to me as I was out shopping the other day. I'm a pretty value-conscious consumer, always trying to find the best deal on products that I buy. You'll never catch me supporting all those TV commercials by buying Brand B aspirin, when I can buy the store's Brand X generic stuff for a tenth the price. Likewise, I go for the generic sandwich bags, hamburger buns, and Klee...uh, I mean, facial tissues. (But never the toothpaste; it tastes too awful.)

Anyway, if people can accept everything from generic laundry detergent to generic paperback mystery novels, why not a generic editorial? I hereby declare war on all the verbose, one-sided, self-serving editorials ever produced, and offer the following concise, open-minded, insipid alternative.

It's high time that something be done. The future is approaching, and the past is already gone. The common people and the uncommon, together and separately, live together on planet Earth in harmony and discord. If the situation does not change soon, it may change later, but it may not. And then there will not be another chance, unless we decide to try again.

For too long people have sat idly by, or worked their fingers to the bones, while others have not. Exhaustion, renewal, peace, and war have proliferated. Computers have not only helped, but even hindered the situation. They are at once part of the problem and part of the solution, working both for good and ill within the tightly woven ecological fabric which is unravelling and being re-woven before our eyes and behind our backs. As someone has said, "Is it time yet?"

Nevertheless, in spite of the gravity and levity of the situation, there are many who couldn't care less and others who are deeply involved. Is it too much to ask that people either stand up and be counted or hide behind the facade of the silent majority? Where have we succeeded, and where have we failed; and where have we simply ignored the facts and speculations? These are vital and insignificant questions with which some of us might want to consider wrestling, or postponing until another time.

Those who oppose us in these matters do, however, deserve some credit. Their views, while false and childish, demonstrate some validity and mature thinking. If we should discover that some of our own opinions do not hold water after the test of time has run its course and tried them by fire as through a needle's eye, then we should be proud and humble enough to admit that our opponents may have been either right or wrong. Not to do so would merely cause our own position to lose credibility and become more believable.

I realize, of course, that there are problems and advantages to all this. Where others have succeeded, we may fail, and vice-versa. But this spectre of fame and obscurity should not blind us to the possibility that computers may yet provide unique and universal analyses and deductions which will leave us unchanged through metamorphosis. The Pollyannas and doomsayers, in their convergent differences of opinion, may yet stand back-to-back and see eye-to-eye. The final verdict, predetermined by the weight of circumstances, is not yet in.

Let us not be the ones to stand by in passive lethargy, burning our candles at both ends as the dawning sunset recedes in the approaching gray spectrum of the transient and immutable present.

ATTENTION AUTHORS

SoftSide Publications is actively seeking programs, article and review submissions for the TRS-80®, Apple and ATARI® home computers. This is a chance for programmers as well as users to make some money to help pay for the "computer addiction" and get their efforts out where they can be appreciated.

Programs — *SoftSide* has always been the leader in the field of BASIC software and BASIC remains our specialty. However, with the advent of Disk Version (DV), we can now also offer an outlet for Machine Language and multiple language programs which do not lend themselves to printed versions. Games, utilities and educational software, as well as any other applications for the home computer user are preferred, although we will consider virtually any type of program. Hybrid mixes of articles and programs also welcomed.

When submitting a program, please be sure to include **full** documentation of subroutines and a list of variables, as well as a brief article describing the program.

Reviews — Well written, informed reviews of all software for the systems we cover are a regular feature of *SoftSide.* Reviewers should take into consideration all aspects of a particular software package, from speed of execution to programming creativity to the estimated length of time that the product will hold the customer's interest.

Articles — We welcome article submissions of all types, but prefer those specifically geared to the home computer market. We give our readers information as a first priority, but vary our content to include some humor and commentary.

All text, including documentation and descriptive articles for programs should be typewritten and double-spaced. Extra monetary consideration will be given to articles and reviews submitted on machine-readable media (Scripsit, Super-Text II, etc.). Programs should be submitted on a good cassette or disk. TRS-80® BASIC programs should function under both Level II and Disk BASIC.

Please be sure to pack your cassettes and disk carefully and to include your return address and phone number.

Send to:

SoftSide **Publications**
SUBMISSIONS DEPARTMENT
6 South Street
Milford, NH 03055

We regret that due to the volume we receive, we are unable to return submissions.

Be sure to send for our **Free Author's Guide**. It further outlines the Specifics of our submission procedure.

TRS-80 is a registered trademark of Tandy corporation.

INPUT

From our readers

THE FIRST RESPONSE TO THE MISSILE COMMAND CONTEST IN DECEMBER 1981.

Dear *SoftSide*,

After reading Dean Macy's review of *ATARI®'s Missile Command* game in your December issue I decided to assert myself to the task of being the first to score 1 million points and gain the illustrious, if somewhat vaguely specified prize. My motivation was further sparked by the fact that my wife had recently begun beating me with regularity (at the game) and this would provide the opportunity to regain my self esteem. I had once scored 350,000 points using the "smart missile" option only and I knew if it were possible this was the way to go. I waited for the proper time, my wife studying, our 2 year old, known affectionately as "El Destructo" was finally asleep and I could concentrate on the task at hand. I will spare the details of my assault on the summit save to say that it did take incredible concentration for more than an hour with only minimal time outs. (Hitting the space bar to pause). As I approached the magical goal my mind wandered to thoughts such as, "Where is the Polaroid Camera and is there any film in it," to "Wonder what the prize will be?" The overriding theme was that once and for all I could lose to my wife on a continuing basis without concern. I could forever point to this achievement with pride and state that since achieving this pinnacle the game held little challenge for me. What glory! And there it was! One more Smart Bomb and I was home free. OOOOOO"***"

What cruel twist of fate! I was crushed. The game doesn't record a score higher than 999,999. Why hadn't I played it safe and taken a picture at that point — it would have been easy to pause, snap a picture and go on but in my haste I just kept on going!! Curses!!

All sorts of thoughts raced through my mind. Maybe it was a setup! Maybe Dean Macy knew 1 million would never show up on the screen — no wonder the prize was so vaguely specified! I was mad! I could get even by running the score up to 999,999 then sending in a picture of "El Destructo" next to the score with some wise commments about it being easy or that the 1 millionth digit didn't appear on the film.

But also I realized the futility of it all. My dreams were crushed, there was simply no way I could prove that I honestly did score 1,017,852. My wife refuses to believe it even now, and I appear doomed to despair.

And so, "Ed"' that is my story. Like so many others my accomplishments may never be acknowledged. There is a certain amount of inner pride mixed with feelings of why did I ever waste my time in such an unfruitful endeavor. But it seemed so glorious at the time.

Perhaps you could warn others, save them the heartache, the depression, not to mention the anxiety and despair. The only thing I would really like answered is did anyone know that 1 million was not possible when the article was published. Thanks for your consideration and patience.

Buzz Taylor
Latrobe, PA

Dean's Reply: I enjoyed your letter. I could feel both the excitement and despair you experienced through your good use of verbiage.

We were told by the ATARI® people that *Missile Command* would indeed display all points up to one billion; however, no one here had the time and inclination to prove ATARI®'s boast.

For your assistance in proof testing *Missile Command* I have enclosed your reward in the form of a $15 Software Certificate which can be used for almost anything to fulfill your dreams. (Had you sent a photograph showing a score of 1,000,000+ points I would have known you cheated.)

I hereby acknowledge that you, Buzz, did indeed score over one million points on *Missile Command*. Put away your anxiety and despair. You are a sung hero at *SoftSide*.

***SoftSide* review contest letters usually are not listed in our reader input section (Yours is so good, however, we decide to print it.) I hope you can be satisfied with this letter and prize only.**

Thank you, Buzz, for your honesty.

A LETTER SENT TO LANCE MICKLUS

Dear Mr Micklus,

Your series in **SoftSide** has pretty effectively scotched any ideas that I had entertained of switching my hobby from fun to profit. I guess I'll stick with my 073D-1600 job, programming IBM dinosaurs for the Army. On the other hand, from my experience with *The Mean Checkers Machine* I'm sorely tempted to go on the tournament checkers circuit. I've never considered myself to be more than an occasional player, so I was quite surprised (disappointed!) To find that I could beat TMCM so easily. Perhaps it's just that I have an early version (T1.3). I've also found that the code does not totally agree with the documentation (a situation I've gotten myself into, quite a few times.) Specifically the IQ is initialized to 3 (not 2) at start-up, and, the program does not prompt for subsequent jumps in a single jump situation. I haven't had the occasion to checkout the tape save & restore routines, but everything else seems to be working as stated.

Regarding your comments on pride in one's work (part 4 of your series.) I believe another term would be "craftmanship." I know it goes against the current wisdom of "Team programming," but any code I write in COBOL, BAL, BASIC, or whatever, has my name on it. If a bug shows up, I'll do my best to fix it, even if I'm no longer assigned to that project. I guess that qualifies me as a genuine anachronism.

But enough rambling — Thanks for a good program, & an informative series.

Kenneth E. Thompson
Saint Louis, MO

THE SENSUOUS WHAT??

Dear *SoftSide*,

I know that there is a big prize waiting for the identifier of "J" in the *Sensuous Programmer*, I thought I would take a chance.

The obvious choice, since I don't know any of you, is to run down the list of "culprits" on page 2. "J" could be Jon, a just too obvious choice. "J" could also be Jean, only because the customer service rep has a need, occasionally, to be sensuous. Perhaps Jerry, because anyone at the bottom of the list has ambitions, and, well, you know.

My guess — "Jimmy" Carter — for several reasons.

1) He disappeared about a year ago.

2) He has a good reason to change his name.

3) Rumor has it that he has been writing — but won't tell anyone what, or even if it is worthwhile.

Now that I have guessed your little secret, you can deliver my new DECwriter with the 25 year supply of paper to my home. I'll be waiting!

Stanley E. Rulapaugh
Phoenix, AR

Editors Reply: Sorry but your guess of Jimmy Carter is wrong! And as for the DECwriter with the 25 year supply of paper if you find a contest with that as the prize let us know because we'd all be interested...

A LETTER ADDRESSED TO "J"

Dear "J",

When you announced that you would no longer print letters which contained "systemist" slurs, I felt that this was a very intelligent move on your part. However, don't you think it would be a good idea to refrain from printing them in your articles as well? I am speaking of the March installment of the *Sensuous Programmer*, which contained one of "J's" typical slurs staining her otherwise excellent column: "But then, why would you want an ATARI® if you weren't going to exploit all its graphics versatility?" Is "J" so jealous of the ATARI®'s graphics that she imagines the computer to be of no use anywhere else? I have friends who own both Apple and TRS-80®'s, and although they are both excellent computers, I do not see any field in which they are so superior as to justify that statement.

By the way my entry into the *Guess the identity of the Sensuous Programmer* contest is Jerry Butler.

Daniel Davidoff
Brooklyn, NY

'J" Replies: You wrote in reference to the statement in my March column, "But then, why would you want an ATARI® if you weren't going to exploint all its graphics versatility?," calling it one of my "typical slurs."

I can assure you that any slurs which you may have perceived in my columns have not been intentional. I have great respect for all three of the computers I use. At the same time, I recognize the strengths and weaknesses of each system, and when one computer does something with greater (or less) ease than the others, saying so is not a slur.

In re-reading the statement you quoted, however, I can see how it could have been interpreted as a backhanded remark, even though it was intended as a compliment. Please accept this rephrasing: "But then, if you have an ATARI®, it would be a shame not to exploit all its graphics versatility."

PRAISE AND PROBLEMS

Dear *SoftSide*,

I have purchased every back issue I could find in Denver. My family enjoys your magazine so much that I have included a card for all back issues available that we do not presently own. Keep up the good work.

The primary reason for not buying the *DV* or *CV* is that it would take away the educational & learning value derived from typing and debugging programs. I do not subscribe to magazines, because it gives me an excuse to visit local computer stores, but I am experiencing difficulty finding copies in Denver.

Why do some of the program lines exceed three lines of data?. My ATARI® 400 is unable to handle four lines as is found in many programs listed in your magazine.

Don Carson
Golden, CO

Editor's Reply: We are trying to get the magazine distributed to as many stores as possible. If there are stores in your area where you think *SoftSide* should be found, send us the names and addresses; we'll be glad to contact them. As for your problem with the program lines, they have to be entered without spaces and using abbreviations for the BASIC keywords. For abbreviations see the appropriate list in the ATARI® manual.

HINTS & ENHANCEMENTS

From our readers

OLD BUSINESS, NEW BUSINESS

I have been wanting to write for a long time, but have never got around to it, so please excuse the length of this letter, as it takes care of a large number of things, going back quite a while.

Starting in chronological order, the disk update times given in the article "DOS" by Lance Micklus, in the September, 1979, *PROG-80*, are incorrect. Not that they don't function, but in changing the stepping times, he is removing functions that the same commands take care of, which explains his errors using APF (re: Old Business, February, 1980, *PROG-80*). Correct stepping times are given below, along with both POKE addresses and track, sector, byte for SUPERZAP. In addition, a ZAP address is given for changing the stepping time in the boot sector, as well as a BASIC program if you don't have SUPERZAP.

For the majority of you that have not read the article, it is possible to speed up the stepping of the head from track to track, which consequently speeds up disk access time. By POKEing the given values into the given addresses, or using SUPERZAP if you have it, you can speed up the 40 millisecond access time used by the operating system.

If you have very, very old Radio Shack drives, you must keep it at 40. If you have non-Radio Shack drives, see the specifications; otherwise, you can run your Radio Shack drives at 20 mS. If you have the newest Radio Shack drives, purchased after January, 1981, you can run at the almost stripping speed of 5 mS, which is what I do. These are the same drives as used for the Model III, which runs at 5 mS and 40 tracks.

In the article "TRS-80® FORTRAN Subroutines" in the August, 1980, *PROG-80*, ICODE = 1 in line 300 of POINT should be changed to ICODE = 0, and ICODE = 1 should be changed to ICODE = -1 in the next line to correspond to TRUE and FALSE.

COPYCAT+ in the November, 1980, ***SoftSide*** and LIVEKEYS in the May, 1980, issue, designated "The World's Smallest Word Processors," are badly written and horribly inefficient. Below is a TWO-LINE program that performs all of the functions of *COPYCAT+* and more efficiently to boot, replacing the ridiculous, arbitrary, and silly "#" for a backspace with a simple backspace! Perhaps Mr. Werbeck's thinking in repetitive loops, duplicating entire sections of code unnecessarily, when he could have built it around the simple statement B$ = A$ causing a rotation, caused him to miss the fact that you can test for a backspace! The cursor in my program is the real cursor, not a badly handled program-defined one.

If you add the PRINT CHR$(14);: to the beginning of line 20 in my program using the ? in the EDIT mode, which will cause the line to go over 255 characters and LIST badly, then type DEFSTRA-Z:CLS:GOTO20, you can make this a loose one-liner. Although this is a one-line program, it is still not the world's smallest word processor. That honor is held by the following program:

```
0LPRINTINKEY$;:GOTO
```

It even offers printer functions such as underlining and extended character modes using the shift-down arrow control function, which even the *Electric Pencil* doesn't give.

In reference to the "Hiding Your Code" series, a much simpler way is to simply

```
POKE16863,195:POKE16864,25:
POKE16865,26
```

which will completely disable LIST. POKE16863,201 to restore.

That brings me to another point: string packing. If I saw "the slashes are POKEd with a Machine Language program SO SAVE IT BEFORE YOU RUN" again I think I would be sick! The idea of packing a Machine Language routine into a string in a program is so that you can leave it there, remove the DATA statements and POKES (most of them), and save string space and initialization time at the same time. RUN it, then remove the packing section (leaving the USR initialization), and THEN save it.

The *MODPROG* program in the August, 1980, ***SoftSide*** and the *AUTO MODPROG* in the April, 1981, ***SoftSide*** will not work as shown. If a byte in a line number or line pointer corresponds to the byte being searched for, the line number will be changed, causing improper functioning, or the line pointer will be changed, causing the computer to lock up.

"Adventures," listed in the index on page 68 of the May, 1981, issue, actually went in the first issue on the next page. Look carefully at my correction to *Kidnapped* in the Bugs, Worms, etc., section of the June, 1981, ***SoftSide***, especially the first line. This is for the "MAKE" verb. The last three letters of the noun are in E$. I wonder what room 37 is... Hmm...

Finally, sound. Everywhere I see "use an amplifier." You do not need an amplifier, as you have a nice tape recorder sitting right there next to you, with the AUX plug already plugged in. If you're just interested in hearing the sound, not having it sound real, just plug your earphone into the EAR jack and press RECORD and PLAY at the same time. If you want good quality, plug a speaker with a phone plug into the EAR jack or find a patch cord. I have an eight-inch speaker hooked up through a phone-to-RCA female cord.

John S. Owens
Virginia Beach, VA

Editor's note: Mr. Owens' comment about saving programs which use packed strings is a good one. But we still advise you to SAVE the program before RUNning it, until you are certain that it's thoroughly debugged; and then to keep a final, pre-RUN version so that modifications may be made more easily later.

Following is the chart and BASIC program relating to disk stepping speeds:

DOS	T,S,B	ADDRESS (HEX)	STEPPING SPEED			
			40mS	20mS	10mS	5mS
NEWDOS	0,7,0F	17484,444CH	0B(11)	0A(10)	1D(29)	08(8)
	0,9,0F	18007,4657H	1B(27)	1A(26)	19(25)	18(24)
TRSDOS	0,7,52	18007,4657H	1F(31)	1E(30)	1D(29)	1C(28)
	0,7,C5	18122,46CAH	0B(11)	0A(10)	09(9)	08(8)
BOTH(BOOT)	0,0,BA		1B(27)	1A(26)	19(25)	18(24)

1 CLEAR1000: OPEN "R",1,"BOOT/SYS.F3GUM": FIELD 1,255 ASA$: GET 1,1: B$ = A$: MID$(B$,187,1) = CHR$(26): LSETA$ = B$: PUT1,1: CLOSE: 'Change the 26 to the value in the table (24-27).
2 '(I decided not to give them the real password for BOOT/SYS, Randy. Who is she, anyway?)

And here's the two-line word processor:

10 DEFSTR A-Z: CLS: PRINT CHR$(14);
20 A = INKEY$: IF A = "" THEN 20 ELSE IF A = CHR$(8) THEN B = "": PRINT A;: GOTO 20 ELSE PRINT B;: B = A: IF A = CHR$(13) THEN LPRINT: B = "": PRINT CHR$(138): PRINT CHR$(30);: IF PEEK(16416) + 256*PEEK(16417) > 15808 THEN POKE 16416,0: POKE 16417,60: PRINT CHR$(30);: GOTO 20 ELSE 20 ELSE PRINT B;: GOTO 20

WHICH WAY IS UP?

I thoroughly enjoy being the aircraft commander in William J. Edmunds' *Instrument Flight Simulator* program (December, 1981). I do, however, experience vertigo on occasion when rolling out of a 90 degree bank or flying inverted, because there is no reference as to "which way is up."

To solve this problem, I added two vertical tick marks, one on each end of the artificial horizon on the attitude indicator. The tick marks point toward the ground. To make the change in the program, the following statements should be changed or added as shown:

```
230  SCALE= 4: ROT= R
4070  SCALE= 4: ROT= R6
9010  DATA 84,0,91,0,100,0,109,0,
      125,0,127,0,27,231
9090  DATA 59,63,63,63,63,70,73,7
      3,73,41,45,45,45,45,6,0
9095  DATA 4,0,79,23,36,4,0
9100  FOR X = 768 TO 899: READ Y:
      POKE X,Y: NEXT : POKE 232,0
      : POKE 233,3
```

Harvey M. Paskin
Dayton, OH

TRS-80® MODEL II

In my last letter, which you published in the July issue, I said that I would find out how string arguments are handled by the USR function of Level II BASIC on the TRS-80®. I did, and it is a simple but rather indirect system. Naturally, the number type flag is set to 3 for a string variable, and sitting right where a two-byte integer value would be is an address. This address points to a three-byte "string descriptor." This "string descriptor" is exactly like what you find at the address given by VARPTR for a string variable. The first byte is the length of the string and the next two are the address of the first character of the string. I wrote a real fancy version of that address conversion subroutine and had it working. Then my TRS-80® Model II arrived and I have been neglecting my Model I ever since.

Before abandoning my Model I, I did find an even simpler method of substituting functions selected at run time in a numerical expression. Write the expression using the USR function as the dummy function and then at run time set the address for the USR function as the address given in *Pathways Through the ROM* for the desired function. This works for most single argument functions.

The Model II BASIC has a function, INPUT$, which has as its argument the number of characters to be input and when that many characters have been typed in, it returns them as a string. The characters are not displayed on the screen which makes it very useful for passwords or other items that you do not want advertised. The following Level II BASIC subroutine achieves similar results using the INKEY$ function. NC% contains the number of characters and the string from the keyboard is returned in KS$.

```
1000 KS$ = ""
1010 KS$ = KS$ + INKEY$: IF
LEN(KS$) < NC% THEN 1010 ELSE RETURN
```

David S. Tilton
Manchester, NH

DELETING BLOCKS

After reading J. Arthur Gleiner's method of deleting blocks of lines in ATARI® BASIC (November, pg. 7), I remembered a program I had seen in some notes on BASIC that Atari sends out. Here is a modified version:

```
1 PRINT "Low, High";:INPUT LOW,HIGH:ST
=20
2 FOR I=LOW TO HIGH STEP ST:GRAPHICS 0
:POSITION 2,2:FOR J=I TO I+ST-1:PRINT
J:IF J<>HIGH THEN NEXT J
3 PRINT "CONT":POSITION 2,0:POKE 842,1
3:STOP
4 POKE 842,12:NEXT J:STOP
```

The program causes the screen editor to go into a "forced read" mode, so that lines on the screen are entered without the carriage return being pressed. Line 4 MUST be a separate line, because "CONT" restarts execution on the next line. To use, just type "RUN" and enter the limits of the range of lines to be deleted. Warning: this program has occasionally caused BASIC to stop functioning (you have to turn the power off and on to fix it), so save your program before running it!

John S. Kalstrom
Goleta, CA

RESCUE MADE EASIER

I have played and enjoyed *Space Rescue* many times. However, I have found that (on the TRS-80® version) it is difficult to maneuver the ship one space at a time. At present, one must hold down the CLEAR button and an arrow key. The following changes allow you to use the down arrow in conjunction with the arrow keys instead of the CLEAR key.

Line	Change
300	98 to 112
305	34 to 48
315	66 to 80

I find this easier to use.

Dan Singer
Washington, DC

SWEEP STAKES

The *Maze Sweep* program for TRS-80® (February, 1982) was great, but it was lacking the routines to keep the ten highest scores on a file. Below are the modifications.

```
25 GOSUB 9100
30 REM DELETE LINE 30
2140 PRINT : PRINT "PRESS <ENTER> TO PLA
Y AGAIN " : PRINT "PRESS 'Q' TO QUIT "
2150 IF INKEY$=CHR$(13) THEN 70
2160 IF INKEY$=CHR$(81) THEN 9010
2170 GOTO 2150
9000 CMD"B","OFF" : RETURN : REM THIS LI
NE FOR MODEL III ONLY
9010 OPEN "O",1,"SEARCH/FIL"
9020 FOR X=10 TO 1 STEP -1
9030 PRINT#1, SC$(X); ","; SC(X)
9040 NEXT X
9050 CLOSE 1
9060 CMD"B","ON" :  REM USE THIS LINE FO
R MODEL III OR
9060 POKE B,B1 : POKE B+1,B2 : REM THIS
LINE FOR MODEL I
9070 END : STOP
9100 REM *** READ IN HIGH SCORES ***
9110 OPEN"I",1,"SEARCH/FIL"
9120 FOR X=10 TO 1 STEP -1
9130 INPUT#1, SC$(X), SC(X)
9140 NEXT X
9150 CLOSE 1
9160 RETURN
```

To initialize the file, enter and run the following program.

```
10 DIM SC$(10), SC(10)
20 FOR X=1 TO 9
30 SC$(X)="" : SC(X)=0
40 NEXT X
50 SC$(10)="JAMES GARON" : SC(10)=6640
60 OPEN"O", 1, "SEARCH/FIL"
70 FOR X=10 TO 1 STEP -1
80 PRINT#1, SC$(X); ","; SC(X)
90 NEXT X
100 CLOSE 1
110 END : STOP
```

Vickie Frederick
Lawndale, CA

OUTGOING MAIL

by Nancy Lapointe

Well another month has passed and this issue is just about ready to go to print. My cue came about a week ago to do my part and write this month's Outgoing Mail. So, here I sit in front of the keyboard generating all the gossip and news tidbits about ***SoftSide*** and its creators.

At this time we would like to extend a welcome to our new Software Manager William Kubeck. Bill will be selecting programs for ***SoftSide*** as well as organizing our programming department.

It was brought to my attention by the Submissions Department that you programmers seem to be holding onto your programs instead of submitting them for publication. What happened to all of you who sent for author's guides? We haven't heard from you! Where are all of you who have had something previously printed? Don't you have the energy, ideas or quality programs to top your last opus? Are all of you too busy using other people's programs to write an even better program for the readers of ***SoftSide*** to enjoy? Maybe there isn't anyone writing anymore. Maybe the thrill of seeing one's name in print has disappeared! Just let it be known, ***SoftSide*** is still looking for good quality programs to publish. Send in those submissions.

All of you CV or DV subscribers and other users of ***SoftSide*** on media will find that starting in this issue we will be publishing loading instructions on the intro page to the sectionalization. We have found that sometimes there are little quirks that make using the media difficult, especially if you don't know how to load the disk or cassette. Now you can check this source of information before thinking, "Another piece of software that won't boot." Here is your chance to avoid unnecessary use of the U.S. Post Office only for us to find that there was nothing wrong with the media. It just wasn't loaded correctly. We're sure this information will help.

Since the survey we have been trying to correct the codes on people's address labels. It really helps us to know what computer each of you is on regardless of whether you are a magazine subscriber or media subscriber. We have been trying to determine what code to give each of you but sometimes it is impossible. It would help to have new subscribers make sure to tell us on their subscription order what computer they are currently using. Also for those of you renewing please mark your renewal notice clearly as to what your code should be.

Here is a breakdown of the codes and their meaning. Your code appears on your address label opposite your name.

TRM — Magazine-only subscriber using a TRS-80® computer.

TRC — TRS-80® computer user with a cassette subscription.

TRD — TRS-80® Mod I computer user with a disk subscription.

T3D — TRS-80® Mod III computer user with a disk subscription.

APM — Magazine-only subscriber using an APPLE computer.

APC — Apple computer user with a cassette subscription.

APD — Apple computer user with a disk subscription.

ATM — Magazine-only subscriber using an ATARI® computer.

ATC — ATARI® computer user with a cassette subscription.

ATD — ATARI® computer user with a disk subscription.

If your code reads anything like OOM it is because we don't know what computer you are using. Please help us out and let us know what your code should be. After sending in your change allow two issues to go by before you expect to see the change. Thanks for your cooperation.

At this point I must break the bad news in reference to the addition of the TRS-80® Color Computer to ***SoftSide***'s list of supported computers. We had projected the addition of the color computer as of the June issue but due to the lack of submissions, the cost of supporting an additional system, and the additional staff required to accommodate the extra work we are now hoping to include the color computer in the fall. It is difficult for me to bring this news to you because we are as disappointed as you, the reader, but rather than do a mediocre job on a new system we feel it best to wait and deliver only the best. We will continue to put our efforts into the systems we are currently supporting thus providing you with a good quality magazine. We have learned that you don't produce anything well when you spread yourself too thin.

As of the May issue we are going to be offering two versions of the TRS-80® Disk for all DV subscribers. If we have an indication on a survey or subscription order that you are using a MOD III system you will receive the MOD III disk. We are going to enclose a note in the mailing of the May disks asking you DV subscribers to indicate if you want the MOD III version, otherwise you will continue to receive the MOD I version. Also any of you who wanted to subscribe to ***SoftSide*** DV for the MOD III and chose not to because you didn't have the two disk drives necessary for conversion this should be very good news for you. It's here!!!! Now you can choose MOD I or MOD III for the TRS-80®.

I think I have shared enough gossip and news for this month. If I write much more I won't have anything left to clue you in on next month. I have been known to be long on words once I finally get in front of the screen thus causing a panic in the paste up of my column. Besides I'm sure you want to get on to the rest of the issue to see what great things we have in store for you this month. We think you will really enjoy all the material we were able to gather. Until next month — Happy Hacking.

Out of the Dungeon

By Allen L. Wold & Fred D'Ignazio

You have entered the home of the most powerful man on the planet. He owns the Chateau, where the alien artifact you have been seeking for the last three years is hidden. He does not know it is there; you have put together the last pieces of the puzzle only a day ago. Your problem is to get him to let you into the Chateau, alone, just long enough to remove the fabled artifact.

You have come to make a deal, and confront No. 2 in his study. His manner is polite but restrained. He has tried to stop you more than once, for reasons you do not understand, as you combed the city for clues to the whereabouts of the artifact.

"You are brave to come here," No. 2 says.

"You dare not kill me in your own house," you tell him. "You would be the prime suspect."

"That is true, but I will kill you shortly, nonetheless."

"I have found the Master Key," you tell him. With the Master Key, No. 2 can become Planetary Director, and you see by his face that he wants that very much.

"I don't believe you," he says. "Lies like that will not save you."

You describe the Master Key to him briefly. It is a document of about 3000 words, detailing an elaborate ritual.

"All *right*," he says. "What do you want in exchange for the Key?"

"One hour alone," you say triumphantly, "in Chateau Celso."

"Impossible," No. 2 snaps.

"Why is it impossible?" you ask. You think, *what has gone wrong?*

"The secrets of the Galaxy Front are hidden in there, and I can't let you get to them before I find them."

"I don't care about the Galaxy Front," you say truthfully. "All I want is the Capellan Stone, which I have reason to believe is also hidden in the Chateau."

"Too bad," No. 2 says. "I care nothing for the Capellan Stone, but I can't let you into the Chateau."

"Come with me then," you suggest. "You can watch to make sure I don't try to find the secrets of the Galaxy Front. All I want is the Stone. Once I have that, I'll give you the Master Key."

"No," No. 2 says. "There are wards and restrictions you don't understand. I can't let you into the Chateau, under any circumstances. If you want Capellan artifacts, you'll have to be content with something other than the Stone. Now give me the Key."

Everything has gone wrong. Can you get out of No. 2's home alive? Dare he try to take the Key from you by force, even though this planet's police would be down on him within hours? You decide to try a bluff.

"I destroyed the Key," you tell him, "but I memorized it first. If you try to hurt me, you'll lose it forever. I have a very weak heart."

Currently there are three kinds of role-playing games available. First, of course, is the traditional human-moderated game, such as *Dungeons and Dragons, Tunnels and Trolls*, or *Runequest*. Then there are the games which are contained wholly within a computer program, such as the venerable *Adventure*, and all the rest of Scott Adam's fine products. Then there are the games which are half and half, in which elaborate tables and charts are contained in the machine, but a human moderator controls the objectives and flow of the game.

Human-moderated games are the most flexible. Players can go anywhere, change their objectives in the middle of a game, and the unexpected is always possible. However, anyone who has looked at the massive rule books for *D&D* or *Chivalry and Sorcery* knows that the amount of material which must be learned before playing is tremendous. A good deal of time is spent rolling dice, consulting tables and charts, and making interpretations. A dedicated moderator, willing to forgo play in order to make the game work, is absolutely required.

On the other hand, the computerized adventure-type games require no work at all. The player can simply explore the cave, the pirate's island, or the asylum, with no concern for dice, combat results tables, wandering monster charts. The problem here is that there is no flexibility. A player cannot go beyond the bounds of the game as designed.

The third form offers the most flexibility, though there are few games of that type yet available. Here all the tedious work is handled by the computer, leaving the moderator free to design an adventure, and the players freedom to play. Only the constraints and rules of the imaginary world are in the computer, not the plot or the objectives of the adventure.

One problem with all types of fantasy role-playing games (FRPs) is that they are primarily restricted to themes of simple action adventure. The player or players wander through a strange world, defeating adversaries, usually by combat, and garnering rewards in the from of treasure or experience points. (A notable exception is the game *En Garde!*, where one need not fight at all, the world is very familiar, and the objective is to acquire status.) This is true whether the background of the game is pseudo-medieval, interplanetary, or contemporary espionage.

This is due in part to inertia. The first great game was *D&D*, and game designers, no less than television producers, imitate success.

The greater reason, however, is that when one is on an adventure of exploration and treasure-hunting, the other non-player characters (NPCs) one meets need be portrayed in only the simplest sense, unlike No. 2 in our scenario above. Even in human-moderated games, where the moderator assumes the role of the NPC, this is the rule.

One of us, Allen, has run a dungeon, and it is extremely difficult, if not impossible, to effectively portray several dozen different NPCs, even if they are only briefly on stage. In computer games, the NPCs have no personality at all.

	Stand	Right	Left	Attack	Retreat
Stand	10%	20%	20%	25%	25%
Right	5%	15%	25%	30%	20%
Left	5%	25%	15%	30%	20%
Attack	0%	25%	25%	40%	10%
Retreat	20%	15%	15%	10%	40%

Figure 1

In an earlier column, we discussed the possibility of board games containing all the tedious tables. leaving the player free to plan strategy and tactics — in other words, to play the game. Fantasy role-playing games of the future will most certainly be like this, what we call Computer-Assisted Role-Playing Games (CARPGs). The advantages are obvious.

Such a CARPG will present the players with a situation, in which there is a problem to be solved, and will provide all the background, all information, the results of any action taken, and all other characters, the NPCs. Our scenario is an example of what such a game might be like.

Eventually, even the necessity for a live moderator will disappear, when the computer can take over the chores of plotting, continuity, and most importantly, the portrayal of NPCs.

We need not concern ourselves with how this will be reproduced graphically or with sound effects. That is another topic, and we will assume for the moment that such a game will be amply provided with both. We mention them now just to indicate we haven't forgotten them.

As we can see, one of the major problems in providing such a game is the creation of NPCs which behave realistically. We can chose plots/themes. The computer can keep track of all the details, role the dice, and enforce the rules. But NPCs with behavior is the sticking point.

By behavior, we mean that the NPC, whether a troll or something like No. 2, does more than just attack, steal treasure, or impede one's passage. The NPC responds to the player, and that response is essentially unpredictable and complicated.

Eliza is a program which plays the role of a psychiatrist, and there are other similar programs. They are all very complex, very long, and require lots of memory. Even such programs are limited, in that their behavior is deterministic. A true NPC's behavior will not be absolutely predetermined, but will make use of random weighted probability tables.

This is a complex subject, but we'll try to present it simply. Let us assume the player has met an NPC. In the most basic form, that NPC has a few "states." A state is what the NPC is represented as being or doing at the moment.

In our example, the NPC has five states: Stand Still, Move Right, Move Left, Attack, and Retreat. The game designer assigns certain probabilities to each state, so that, for example, the NPC will stand still 10% of the time, move right or left 20% of the time each, attack 25% of the time, or retreat 25% of the time. Whatever it has done in the past bears no relation to what it will do next. The chances of any action, the probability of any state, are the same at all times.

More complex behavior is possible, making use of what is called a Markov Chain. In this case, the NPC moves from one state to another, as before, but the probability of any future state is dependent on its present state (or last movement). This is best shown with a table which you can see in Figure 1.

The column on the left represents the state the NPC is in, or, in other words, the action the NPC took the last time. The row across the top represents the states the NPC could take the next time. The percentages in the table, called transition probabilities, represent the probability of the NPC moving to any future state, given a present state. As you can see, this makes the NPC much more interesting.

There is still a higher level of behavior, which is much more complicated. The NPC using a Markov Chain remembers nothing. That is, its state of having moved left the 72nd time is the same as its having moved left the first time. If we provide an NPC with "memory," then we might have the effect that the NPC will move left no more than three times in a row, or if it has retreated once, it will never attack.

By adding to the number of states, and by adding memory to the NPC, we can come up with very complex behavior. We are approaching the realms of Artificial Intelligence, which we will discuss in more detail in a later column.

Another question is, how to construct these complicated tables, not only for character behavior, but for other elements in the environment. Most events do no occur in a truly random fashion, but according to rules which modify chance. (For an excellent discussion of how the principles of nature govern chance, see *Laws of the Game*, Eigen and Winkler, Knopf, 1981.)

To provide a truly believable CARPG, real-life situations must be accurately simulated. This kind of research is, of course, already being done. Business, politics, physics, and other disciplines make extensive use of simulations. In the realm of games, see the massive rules for *Chivalry and Sorcery*. But of all simulations, that of human behavior is most difficult.

Most NPCs will be only met briefly, and need not be very complex. A band of "orcs" will be similar to each other, though each individual will be controlled by its own set of charts. Randomly-met humans can all behave in a more or less typically human way, but there must always be the oddball, and there must be the more important characters. The tavern keeper, the starship captain, the woman with the clue to the mystery, must all be distinct individuals.

The technological and programming skills necessary to provide truly believable NPCs are formidable, though not out of our reach. But there is still a limit to what can be done by people whose business is programming or game design.

What are needed are good story situations and real characters, developed by people whose business it is to provide such — that is, professional authors.

Movies are made from books and

stories. There is no reason FPRs or CARPGs can't be made the same way. This provides a much broader range of theme, plot, character, situation, and so on. Some SF novels or stories which might well be made into FRPs include *Hammer's Slammers,* by David Drake, any of the Kane books by Karl Edward Wagner, *ESPer*, by James Blish, or the Amber books by Roger Zelazny.

In a short story, there is a single objective, and the failure or success in achieving that objective brings the story to a close. In a novel, on the other hand, the major objective is not always clear, and situations arise which require the character to seek short-term objectives. A properly-designed game, the product of the cooperative effort of storytellers, programmers, and game designers, will not have these short-term goals built in, but will be able to provide them as needed. Needless to say, the design, production, and execution of such a game will be extremely complex.

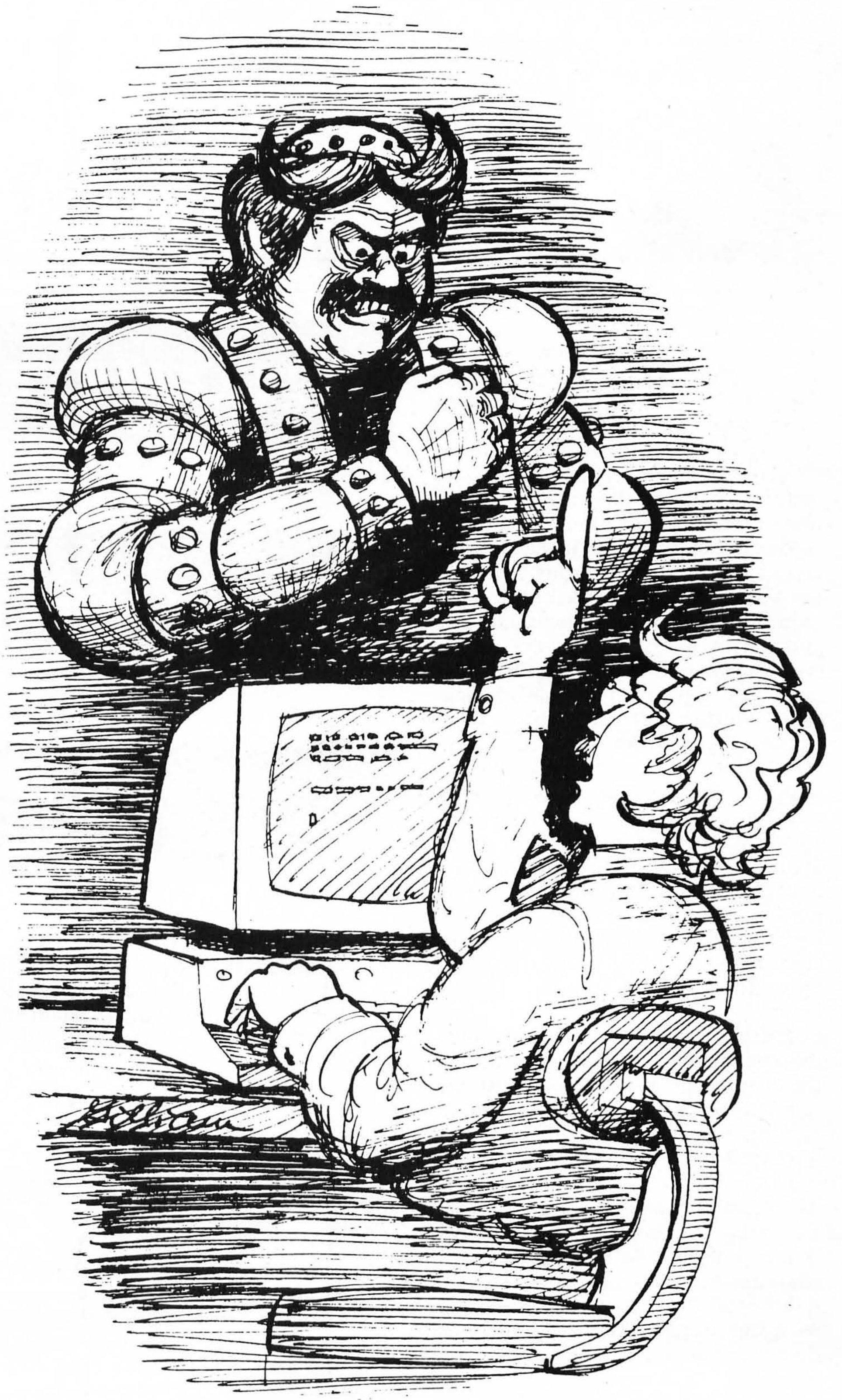
It is just a matter of time before we will see CARPGs with a broad range of general themes, such as mystery, romance, western, soap-opera-like contemporary life, true science fiction, true fantasy. Once a player has chosen such a theme, he or she will then have the ability to choose a prime objective, such as the accumulation of wealth, the achievement of power or fame, discovery, revenge for past wrongs, romantic and/or sexual fulfillment, and so on.

The player will also be able to chose his or her own character, which could ressemble that player in varying degrees, or which could represent entirely different roles. The computer would assist in playing those roles, based on programs written by people who understand psychology and human nature.

At the same time, the player will be able to decide just how difficult a problem will have to be solved, the number of obstacles to be conquered, and the degree of opposition from "nature" as represented by the game's program.

Assisted by the computer, the player will be able to live out fantasies, not just in the imagination, and not in real life, but on a computer. Fantasizing is a problem-solving activity, and with computer assistance, the constraints on the problem can be realistic, and the validity of the solutions checked against reality. Needless to say, the entertainment aspect of this kind of activity would be immense.

This is a long way from the dungeon.



We have only touched on a rich and complex set of ideas, which we hope to develop in more detail at a later time. The point is that fantasy role-playing games, as they are today, are just the beginning. One of the first steps toward achieving the ultimiate, as described by Arthur C. Clarke in *Against the Fall of Night* and *City and the Stars* is the development of truly interactive characters. Work on that has already begun.

Transformation Techniques

by Joan Truckenbrod

Shapes or figures can easily be transformed into new forms with the aid of a transformation program. With this program, shapes or figures can be changed in a systematic way, to create new figures or distorted forms. Many processes involving change, such as ice melting, parallel the idea of shape transformation. For example, the concept of metamorphosis in nature provides a source for numerous ideas that can be translated into interesting visual sequences with a transformation program. This technique can be used to create animated sequences in which one figure appears to gradually change into another figure. Abstract or realistic figures can be used in this transformation program. As shown in figure 1, this program is designed to transform one figure into another in a given number of steps. The artist defines the first figure and the last figure and specifies the number of steps between these figures in the transformation sequence. The program generates the in-between steps and draws the series of figures as seen in Figure 1.

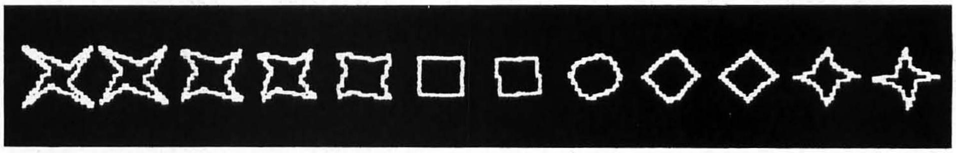


Figure 1

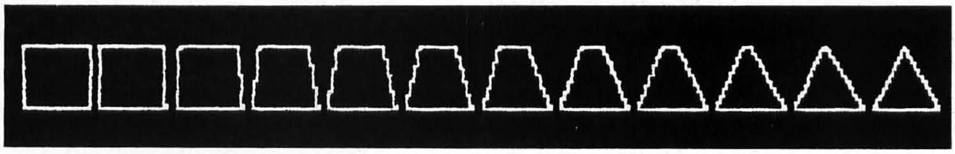


Figure 2

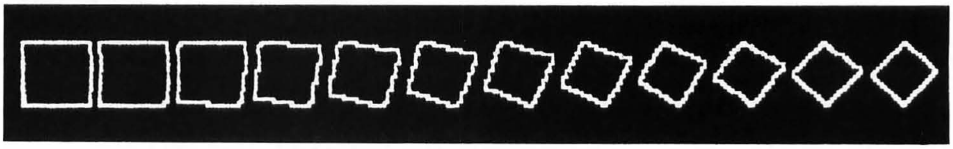


Figure 3

The basic procedure for transforming one figure into another is finding the distance between each pair of corresponding points on the beginning and ending figures, and dividing this distance by the number of in-between steps specified in order to arrive at the increment of change necessary for each step in the sequence. Initially, the two figures must be defined in X and Y coordinates. This program draws a series of twelve figures across the Apple display screen, so the width of each figure (or maximum X value) is limited to 23 units.

Each figure must contain the same number of data points. If one figure actually has less points than the other figure, then points can be double numbered as shown in figure 2. If closed shapes are desired the first point in each shape must also be the last point listed in the data list. The number of points used to define each figure is specified in line 60 of the program as the variable NP. The coordinate values describing the first figure are put into the X1 and Y1 arrays, and the data for the last figure is put into the X2 and Y2 arrays.

Figures are transformed in these examples by gradually changing the X and Y coordinates that describe the initial figure. The increment of change for each X and Y coordinate in the figure is determined in lines 165 through 210 in the program. This portion of the program finds the distance D between each pair of corresponding points in the first and last figures, and divides each distance by NS which is the number of steps in the sequence beyond the first shape. The increments for each point are put into the CX and CY arrays. Lines 250 through 330 repeatedly draw the figure, changing the X and Y coordinates according to the increments in the CX and CY arrays prior to each step in the sequence.

This program can be used to create unique visual effects. Figures can appear to rotate in different directions as shown in figures 3 and 4. This effect is created by using a square as the first figure and a diamond as the second figure. In figure 4 a triangle and an inverted triangle are used as the first and last figures. Using the same figure in different orientations will create the effect of a rotating figure. Another effect to experiment with, using this program, is changing the scale or size of an object or figure. Using the same figure in two sizes for the first and last figure will create the effect that the figure is growing or shrinking. Figures 5 and 6 illustrate these size changes. As shown in figure 6, shapes can be outlines or shaded areas. This program can also be used to transform one letter into another as in figure 7. In addition to letters, realistic figures as in figure 8 can be drawn to create an interesting transformation. Experiment with a wide range of shapes, forms and figures to create interesting compositions or animated sequences based on transformations. This program will be applied to animated sequences in future articles.

```
10  REM  SHAPE TRANSFORMATION
20  REM  BY JOAN TRUCKENBROD
30  REM
40 NS = 10: REM NUMBER OF STEPS D
     ESIRED BETWEEN BEGINNING AND
      ENDING FIGURE
50 NS = NS + 1: REM ACTUAL NUMBER
      OF TIMES X AND Y COORDINATE
     S ARE INCREMENTED FOR THE TR
     ANSFORMATION SERIES
60 NP = 5: REM THE NUMBER OF POIN
     TS USED TO DEFINE EACH FIGURE
70  DIM X1(NP),Y1(NP): REM COORDI
     NATES FOR BEGINNING FIGURE
80  DIM X2(NP),Y2(NP): REM COORDI
     NATES FOR ENDING FIGURE
90  DIM CX(NP),CY(NP): REM INCREM
     ENTS USED TO CREATE TRANSFOR
     MATION SERIES
100  REM
110  FOR I = 1 TO NP: READ X1(I),
     Y1(I): NEXT I
115  REM DATA FOR FIGURE ONE
120  DATA 0,20,20,20,20,40,0,40,0
     ,20
130  FOR I = 1 TO NP: READ X2(I),
     Y2(I): NEXT I
135  REM DATA FOR FIGURE TWO
140  DATA 10,20,10,20,20,40,0,40,
     10,20
150  REM
160  REM THE FOLLOWING LOOP DETER
     MINES THE DISTANCE BETWEEN E
     ACH PAIR OF POINTS AND THE X
      AND Y INCREMENTS NECESSARY
     TO TRANSFORM ONE FIGURE INTO
      ANOTHER IN THE DESIRED NUMB
     ER OF STEPS
165  FOR I = 1 TO NP
170  REM D REPRESENTS THE X OR Y
     DISTANCE BETWEEN A PAIR OF P
     OINTS
175 D = X2(I) - X1(I)
180  REM THE INCREMENT OF CHANGE
     IS DETERMINED BY DIVIDING TH
     E DISTANCE D BY THE NUMBER O
     F DESIRED IN-BETWEEN STEPS
185 CX(I) = D / NS
190 D = Y2(I) - Y1(I)
200 CY(I) = D / NS
210  NEXT I
220  REM
230  HGR : HCOLOR= 7
240  REM THE FOLLOWING LOOP DRAWS
      THE SERIES OF TRANSFORMED S
     HAPES
250  FOR N = 0 TO NS
260 XP = X1(1) + N * 23 + N * CX(
     1)
270 YP = Y1(1) + N * CY(1)
280  HPLOT XP,YP
290  FOR M = 2 TO NP
300 XP = X1(M) + N * 23 + N * CX(
     M)
310 YP = Y1(M) + N * CY(M)
315  HPLOT  TO XP,YP
320  NEXT M
330  NEXT N
340  END
```

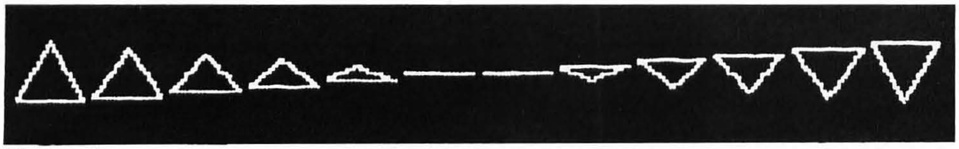


Figure 4

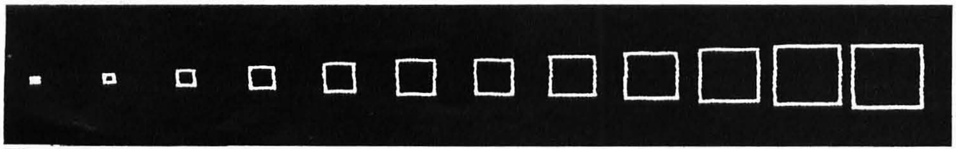


Figure 5

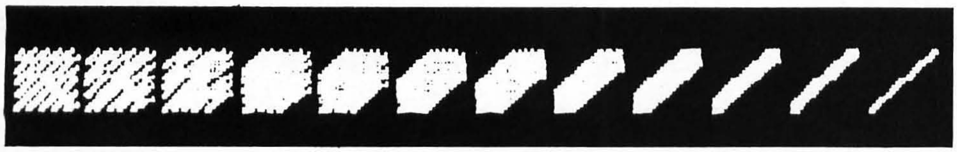


Figure 6

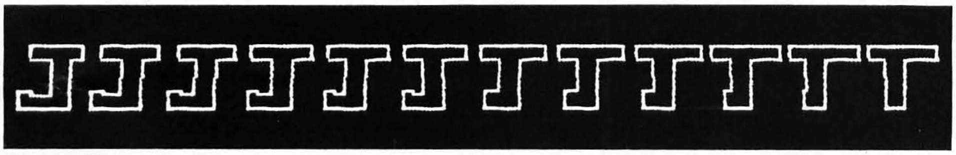


Figure 7

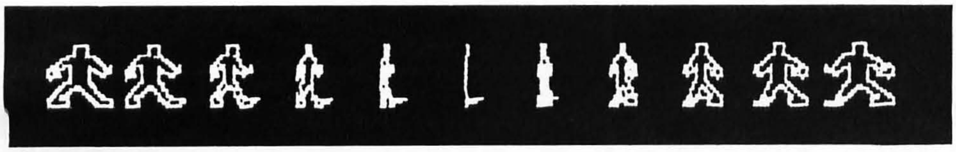


Figure 8

CALENDAR

May 4

Microcomputer Fair
University of New Hampshire, Durham, NH

This fair will focus on educational software for instruction and research and is designed to increase the participant's knowledge about computers and their impact on education.
Contact: Larry LaBelle, Computer Services, Stoke Hall, UNH, Durham, NH 03824

May 6-9

First Annual Washington Home Entertainment Show
Sheraton Washington Hotel, Washington, DC

This show will feature the latest technologies, products and services for home and personal entertainment. Designed for distributors, retailers and manufacturers, the displays will include: personal computers, home video, electronic and video games, video clubs and libraries, home communications and security systems and much more.
Contact: National Trade Productions, Inc., 9418 Annapolis Road Suite 206, Lanham, MD 20706 (301)459-8383

May 14-16

Applefest
Hynes Auditorium, Boston, MA

Hundreds of manufacturers, distributors and dealers will showcase Apple-compatible products including computers, components, peripherals, plug-in cards, publications, gifts and software for home, office and school. Multimedia presentations, seminars and workshops will detail new uses for your Apple.
Show hours are 11 a.m. to 6 p.m. Admission is $6 per day, $10 for two days, or $15 for three days.
Contact: Northeast Expositions Inc., 824 Boylston St., Chestnut Hill, MA 02167 (617)739-2000

May 22

NJ Microcomputer Show & Fleamarket
Holiday Inn North, Newark, NJ

This event will have over 50 commercial exhibitors and 150 fleamarket sellers featuring hardware, software, and accessories for all systems including Apple, TRS-80®, ATARI®, Pet, Heath/Zenith, ZX-80/81, and IBM.
Contact: Kengore Corporation, 3001 Rte. 27, Franklin Park, NJ 08823 (201)297-2526

May 22

First Regional Conference on Technology and Special Education
Mill Neck Manor Lutheran School for the Deaf, Mill Neck, NY

The New York State Association for Educational Data Systems will sponsor this conference for administrators, teachers and parents.
Contact: Dr. Delores Shanahan, Commack Public Schools, Indian Hollow Computer Lab, Kings Park Road, Commack, NY 11725 or Jerry Burke, Half Hollow Hills High School, Dix Hills, NY 11746

May 31

Personal Microcomputer Interfacing and Scientific Instrument Automation Workshop
Virginia Polytechnic Institute, Blacksburg, VA

This is a hands-on workshop with the participants working with and designing interfaces for personal computers.
Contact: Dr. Linda Leffel, CEC, Virginia Tech, Blacksburg, VA 24061
(703)961-4848

If you or your organization are sponsoring or know of an event you think would be of interest to *SoftSide* readers, please send complete information to:

SoftSide Publications
Calendar Editor
6 South Street
Milford, NH 03055

Be sure to include complete information concerning dates, location, subject matter and a contact name, address, and phone number.

SOFTSIDE ORDERING INFORMATION

FORM OF PAYMENT

USA

VISA, MasterCard, certified checks, money orders and personal checks are accepted.

Canada/Mexico

The preferred method of payment is by VISA or MasterCard. A bank check is acceptable if it has been preprinted for payment in U.S. dollars. No personal or company checks accepted.

Other Foreign Orders

Payment must either be by a bank check drawn on a U.S. bank payable in U.S. dollars or by affiliated bank credit cards of VISA or MasterCard.

GUARANTEE

All software is guaranteed to load and run. If you experience difficulties with the product within 30 days, it may be returned for replacement. Send your properly protected tape or disk to the attention of the Customer Service Representative and include your name, address, and the reason it is being returned.

LIABILITY

All software is sold on an as-is basis. **SoftSide** assumes no liability for loss or damage caused or alleged to be caused directly or indirectly by products sold or exchanged by them or their distributors, including, but not limited to, any interruption in service, loss of business or anticipatory profits or consequential damages resulting from use or operation of such software.

PRICES

Prices are subject to change without notice. We are not responsible for typographical errors.

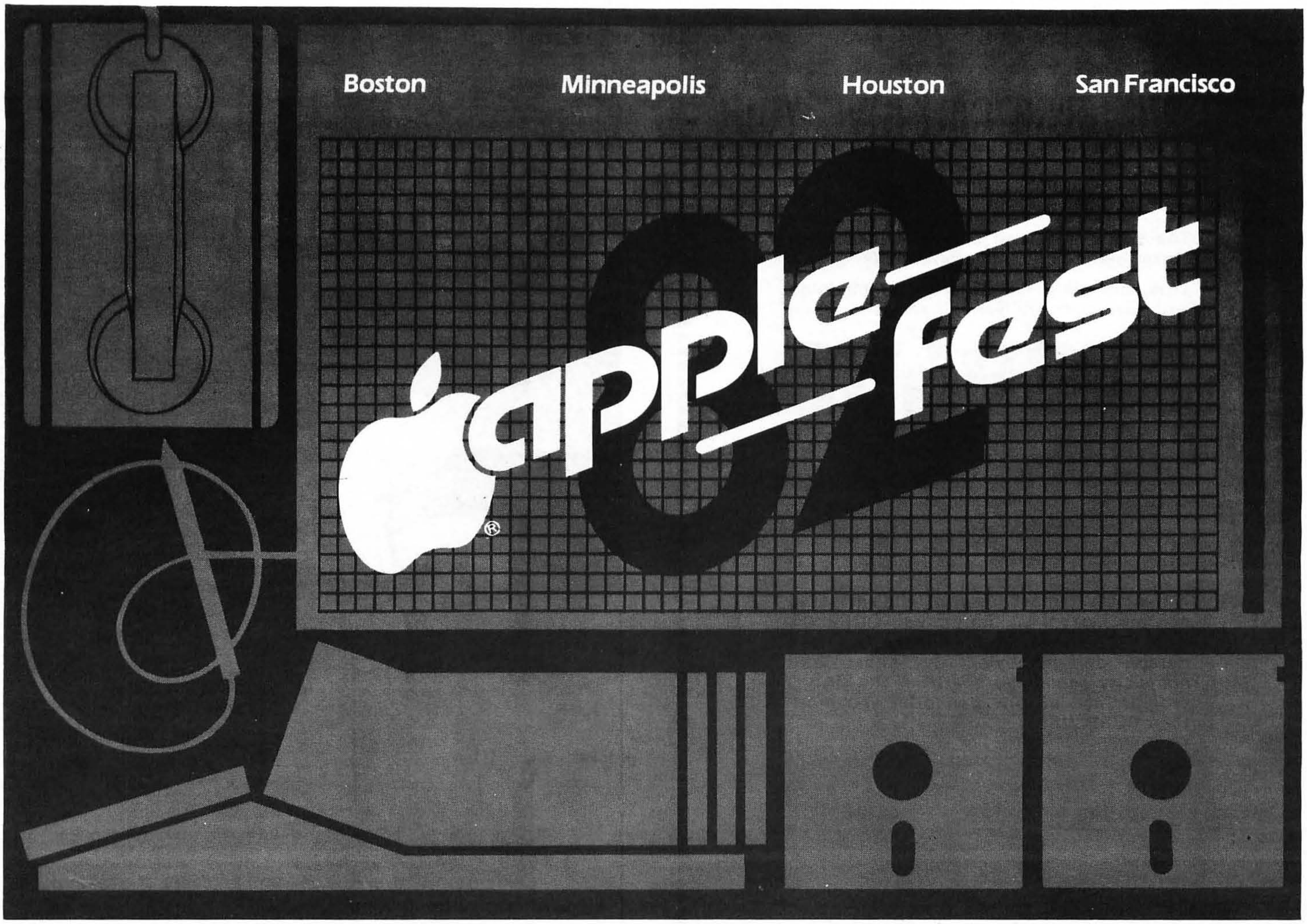
Unless otherwise noted in a published advertisement, the following prices are in effect as of this issue:

	USA	Mexico APO/FPO Canada	Other Foreign
SoftSide Magazine (yr)	$30	$40	$62
CV (year)	$75	$95	$125
(6 mo.)	$39	n/a	n/a
DV (year)	$125	$145	$175
(6 mo.)	$64	n/a	n/a
Adventure of the Month Month (6 mo.)			
Cassette	$29	$35	$41
Disk	$49	$55	$61
Parsec	$20	$21.50	$25
Envyrn	$200	$202	$210
Diversions	$60	$70	$85

BACK ISSUES

Minimum order for magazines only — 3 issues. There is no minimum order for magazine/media combinations.
Price includes shipping to the 48 states only. Alaska, Hawaii, Puerto Rico, APO/FPO, and ALL foreign orders — postage is additional.

ALL Foreign orders and all magazine/media combination orders — Order directly from **SoftSide, 6 South St., Milford, NH 03055.**



The Most Spectacular Extravaganza Ever... For Apple Users

At Applefest '82 hundreds of manufacturers, distributors and dealers will showcase the entire spectrum of Apple-compatible products including computers, components, peripherals, plug-in cards, publications, gifts, magazines, services, accessories and software for home, office and school.

Hands-on centers and multimedia presentations will demonstrate the newest applications for business, education and entertainment.

Seminars and workshops, conducted by the world's leading Apple authorities, will detail new uses to make your Apple more enjoyable and more useful than you ever imagined.

You'll meet thousands of other Apple owners and find the newest of everything for your Apple under one roof . . . and for sale at super show prices.

So if you use an Apple . . . or are thinking about buying one, you won't want to miss a minute of Applefest '82.

Ticket & Hotel Information

Send your check and a note indicating the specific show you wish to attend. Tickets and hotel information will be mailed back to you. Tickets can also be purchased at the show. Make all checks payable to Northeast Expositions Inc. 824 Boylston Street, Chestnut Hill, Mass. 02167 Tel: 617 739 2000.

Exhibitor Information

For specific exhibitor information on one or all of the Applefest '82 shows call Northeast Expositions at the telephone number above.

Applefest/Boston
Fri-Sun May 14-16, 1982
Hynes Auditorium
Show Hours: 11AM to 6PM Daily
Admission: $6 per day or $10 for 2 days, $15 for 3 days

Applefest/Minneapolis
Thurs-Sun Sept 16-19, 1982
Minnesota Auditorium and Convention Hall
Show Hours: 11 AM to 6 PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days, $15 for 4 days

Applefest/Houston
Fri-Sun Nov 19-21, 1982
Albert Thomas Convention Center
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

Applefest/San Francisco
Fri-Sun Dec 3-5, 1982
Moscone Center
Show Hours: 1PM to 10PM Daily
Admission: $5 per day or $8 for 2 days, $12 for 3 days

Applefest is produced by Northeast Expositions Inc. and is sanctioned by Apple Computer Inc. and The Boston Computer Society.

*Apple and Applefest are registered trade and service marks of Apple Computer Inc.

SAVE BY COMPUTERIZING YOUR BUSINESS

by Edward Ting

Cost of computer system	$3500
Cost of programs (inventory, mail list, etc.)	$500
Desk for system	$150
Book to shove under short leg of wobbly desk	$1.95
Ax	$15
Carpet cleaner fee	$50
New desk	$500
Add-on computer room	$4500
Lights for room — 60W	$10
Lights for room — 150W	$10
Sunglasses	$15
Electric bill	$100
Synthetic plush carpet for room	$200
Cost of computer repair from static shock	$400
Anti-static mat	$50
Do-it-yourself high speed modification	$50
Soldering iron	$10
Solder	$1.95
New motherboard	$700
Super Star Trek XXII	$25
Color monitor	$400
Joysticks	$50
Sound amplifier	$20
New Hi-Res *Super Star Trek XXIII*	$50
Wide-screen projection television	$1000
New super-glide joysticks	$100
Stereo system	$1200
Company bankruptcy settlement	$700,000
TOTAL	$713,618.90
Savings from first year of computerization	$625
Net cost of computerization — first year	$712,993.90

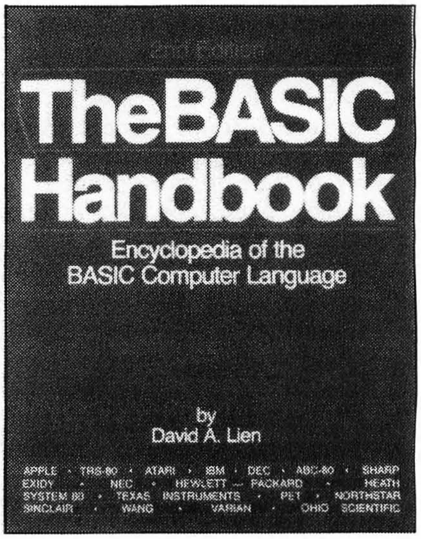
The BASIC Handbook

(2nd Edition)

A Review by Jon Voskuil

By David A. Lien. Compusoft Publishing, 1050-E Pioneer Way, El Cajon, CA 92020. Retail price: $19.95.

Anything you need to know about the meaning of a BASIC keyword will probably be found within its covers.


2nd Edition
TheBASIC Handbook
Encyclopedia of the BASIC Computer Language
by David A. Lien
APPLE · TRS-80 · ATARI · IBM · DEC · ABC-80 · SHARP
EXIDY · NEC · HEWLETT — PACKARD · HEATH
SYSTEM 80 · TEXAS INSTRUMENTS · PET · NORTHSTAR
SINCLAIR · WANG · VARIAN · OHIO SCIENTIFIC

"I have an ATARI® and am trying to convert an Apple program that uses Hi-Res graphics. Could you tell me what the different Apple graphics statements do?"

"I'd like to see a series of articles explaining how to translate ATARI® programs to run on my TRS-80®. The way strings are handled has me completely mystified."

"I've never seen the ampersand used in an Apple program line, and my manual doesn't say anything about it. Is it a mistake, or what?"

One thing that our recent reader survey has made clear is that the majority of you appreciate ***SoftSide***'s multi-system format, and have a desire to learn more about the "other" computers we support. It's high time all of us realize that we are destined (or doomed) to live among legions of computers, each brand and each model just a little different from the others. Your next Atapple-80 will probably differ more from your present model than all those "other" systems now do.

In the midst of such compu-babel stands David Lien's classic book, *The BASIC Handbook*, now in its second edition. Reviewing this book is very much like reviewing Webster's dictionary or Roget's thesaurus. You just sort of take it for granted (which is one of the best recommendations that any product can have). You have it on the shelf; it's there to use when you need it; you depend on it to give accurate information. Its subtitle is a good one: "Encyclopedia of the BASIC Computer Language." Anything you need to know about the meaning of a BASIC keyword will probably be found within its covers.

This is not a programming tutorial; it is a reference book, for use by those who already have a working knowledge of BASIC. In alphabetical order, each BASIC keyword (from virtually every BASIC dialect in existence) has its own entry. Entries are typically one or two pages long, but may extend to several more pages for words such as DIM and PRINT. In general, each entry contains the following types of information:

1. The word itself, along with abbreviations and alternate spellings.
2. What type of word it is (command, statement, function, or operator).
3. A description of what the word does.
4. A test program with sample run, which you can use to test any computer's reaction to the word.
5. Hints on how a programmer might make best use of the word.
6. What to do if your computer doesn't have the word and you want to duplicate its function.
7. A cross-reference to other words for more information.

An 18-page "Index and Scorecard" at the back gives page references for each of the words, and provides blank spaces next to each for indicating a test run pass/fail and other notes for your system. In addition there are brief special sections on the Acorn ATOM, ATARI®, Textronix, and TRS-80® Color computers, outlining some of their special words, and another short section covering the basics of DOS and Disk BASICs.

This second edition is "greatly expanded and revised" over the first (it's 480 pages long), and at a bit higher price. It fills in some of the gaps left in the first, as well as adding many words from new dialects, and attempts a fuller (but not complete) documentation of Disk BASIC usage. With a 1981 copyright date, it is three years newer than the first edition, and will undoubtedly need updating itself in a few years to keep up with the times.

Should you buy this book? Well, it's an indispensable reference work for anyone who works with multiple computers. Or, if you're the kind of person who likes to open up an encyclopedia once in a while and just start reading, then this is definitely for you. On the other hand, if you're not terribly interested in (or involved with) variations in BASICs, then you would have very limited use for it; your own computer manuals would contain most or all of the information that you'd find here.

Perhaps I should put it this way: If you feel that you really do need "something like" this, then — well, this is exactly what you need.

REVIEW

Data Impact Printer Model 84G

By Dean F.H. Macy

from DIP, Inc. 745 Atlantic Avenue, Boston, MA 02111. Basic unit price: $795.

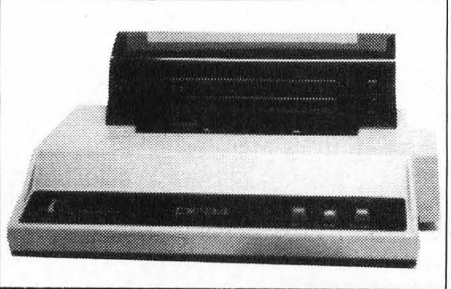
I'm sure ***SoftSide*** readers are aware of the tremendous improvement in the clarity of screen-shots printed in the last five issues of the magazine, especially those of the TRS-80® and, more recently, the Apple. I could say that the improvements came about because of my excellent photographic techniques, but alas, many of the screen-shots you have been raving about are not photographs at all, but screen-dumps. (Oh, no! The truth is out at last.)

What is a screen-shot? Some readers may have the image of a very frustrated programmer holding a 45 Magnum to a CRT and blowing it away. (The idea has tempted me at times.) This is not the case. A screen-shot develops from holding a camera to the screen of a CRT and taking (shooting) a photograph of the information displayed there.

On the other hand, a screen-dump uses a program allowing the computer to "read" pixel information displayed on a CRT screen, translating that information to printer format, and then transmitting the signal to a printer capable of writing pixel by pixel translations. Since the screen-dump writes the identical information shown on a CRT to paper, publication of screen-dumps would be superior to screen-shots.

You may ask, "Why hasn't ***SoftSide*** used screen-dumps before now?" Actually, we have dumped TRS-80® screens to paper for several months using an Epson MX-80 printer and a program written by Alan J. Zett. Starting with the last issue we used a new, superior program which reads pixel information displayed on the screen. Using the *Screen Print* program (included in last month's issue) we can mix text and graphics in a screen-dump. Having screen-dump software, we needed a high quality dot-matrix printer, capable of translating pixel information to paper. Enter DIP.

I became acquainted with DIP several years ago while evaluating products designed for OEM (Original Equipment Manufacturer) markets. I evaluated a DIP 81 printer which I later purchased at some ridiculously low price, and christened "Dippy Printer" after my review was completed. I kept the Dippy Printer, perhaps as a reminder that good packaging can effectively conceal poor design. But that was two years ago, and as computer design has improved, so has peripheral design.



"The DIP model 84G/FT is a low-cost, fan-cooled, high-reliability impact printer. It utilizes ordinary paper in fanfold form and loose sheets, and can be switched from tractor to friction form feed."

As of this writing, DIP, Inc. is one of the leading suppliers of printers to the OEM market. Their regular accounts include such biggies as Hewlett-Packard, Data General, and Digital Equipment Corporation: companies who would never settle for less than the best, at any cost.

In the fall of 1981, DIP, Inc. offered its line of printers to the consumer market, at which time we purchased, for evaluation, an enhanced DIP 84G/FT dot-matrix printer. We had six models from which to choose, including the now-famous (but no longer dippy) DIP model 81, a friction-feed printer with standard 800-byte buffer retailing for $449. (The DIP 81A has a 1K buffer and uses "Control Y" for Telex operations. It retails for $499. Next in line is the DIP 82 which is similar to the DIP 84G except for the lack of a tractor feed and costs $695.) The DIP, Inc. big guns are Models 84G, 85, and 95, and it is this type of printer that I will cover in detail.

The DIP model 84G/FT is a low-cost, fan-cooled, high-reliability impact printer. It utilizes ordinary paper in fanfold form and loose sheets, and can be switched from tractor to friction form feed. Complete with electronics, this printer is designed to interface directly with mainframe, mini, and microcomputers in one of three dip-switch-selectable ways: (1) Using a Centronics-compatible parallel cable; (2) an RS-232C serial cable; or (3) a 20MA serial loop. (Serial baud rate selectable between 50 and 1200 baud. Optional 9600 baud.)

Electronic control circuitry for the entire printer is on a single printed circuit board mounted in front of the printer mechanism. Quick connects enable the board to be easily removed for service or replacement. A Z-80A microprocessor and two Z-80 PIO chips control logic, input/output, and head alignment functions of this computer based printer. Utilized as buffers and font control are 2716 EPROMS. (Our enhanced version uses 2K buffers and a 7x7 font.)

The DIP84G/FT incorporates a dot-matrix impact printer mechanism which utilizes a printing element consisting of seven print solenoids and print wires. The print wires are arranged vertically and are driven within the printing element at a constant speed by clocked pulses to the solenoids. By energizing the solenoids at the appropriate time, as the print head traverses the print area, any desired character may be formed.

In the software-programmable graphics mode the print head moves unidirectionally from left to right when printing each 512-byte graphics line. Plot densities, corresponding to the format of .015 inches between horizontal dots, can be varied in four steps between 33.3 dots/inch and 66.6 dots inch. For continuous graphics (such as the ones at left) the line feed will advance the paper six vertical dots after printing each graphics line.

What does all this graphics stuff mean to the user? Very simply, if you are able to generate a computerized photograph of a person's face on your CRT, the graphics capabilities of the DIP 84G/FT, under software control, could print the image you see on the screen to paper.

These screen-dumps from Mark Cross' *Bronto in the Swamp*, and the commercial game, *Apple Invader*, will give you some idea of the highly detailed scenes which can be created using this versatile printer.

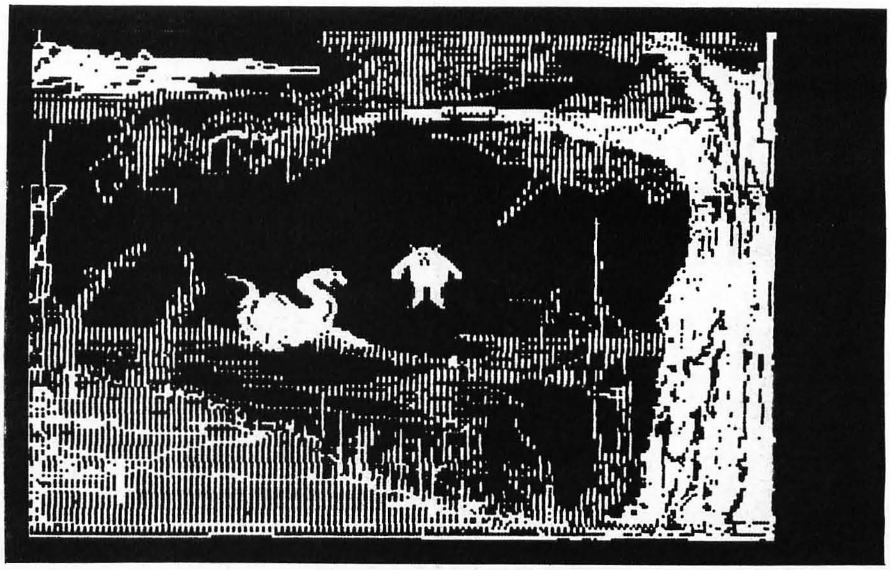
Of course, any dot-matrix printer is useless without the controlling software. If you are an Apple computer user who intends to purchase a DIP 84G printer, be assured that software is available on disk directly from DIP, Inc. for $20. The multiple programs by John Wang allow the user to print positive or negative Hi-Res graphics in single or double resolution from the HGR1 Apple screen.

If your intended dump is from the HGR2 screen, a program must be written to move the image from HGR2 to HGR1 before the dump. Also no Lo-Res program is incorporated on the disk. (At the end of this review the programs we use for Lo-Res and HGR2 dumps are listed.) At this time I know of no software available for dumping ATARI® graphics using a DIP 84G, however I expect one of our readers to send a program to do this in the very near future.

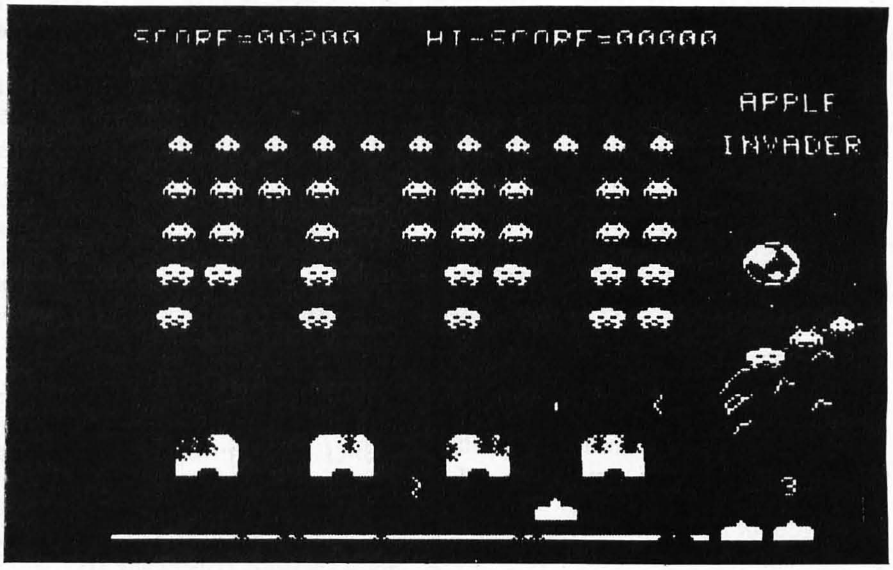
Whereas one may purchase a DIP printer for its graphics capabilities, the purchaser will use a printer mainly for dumping text.

DIP models 84 and 85 ($795 and $895) do not use true descenders however the letters are well formed and cause no text reading problems. The DIP 95, listing for $929, has all the features of the DIP 84G plus the features found on more expensive word processing printers, true descenders included. The DIP 95 and DIP 85 character matrix uses an 11x9 dot pattern, and can be changed quickly to accommodate other sizes (i.e., 9x9, 7x7, 11x7, 7x9), by switching a single chip. The DIP 85 has x-on and x-off features and prints at a baud rate of 9600.

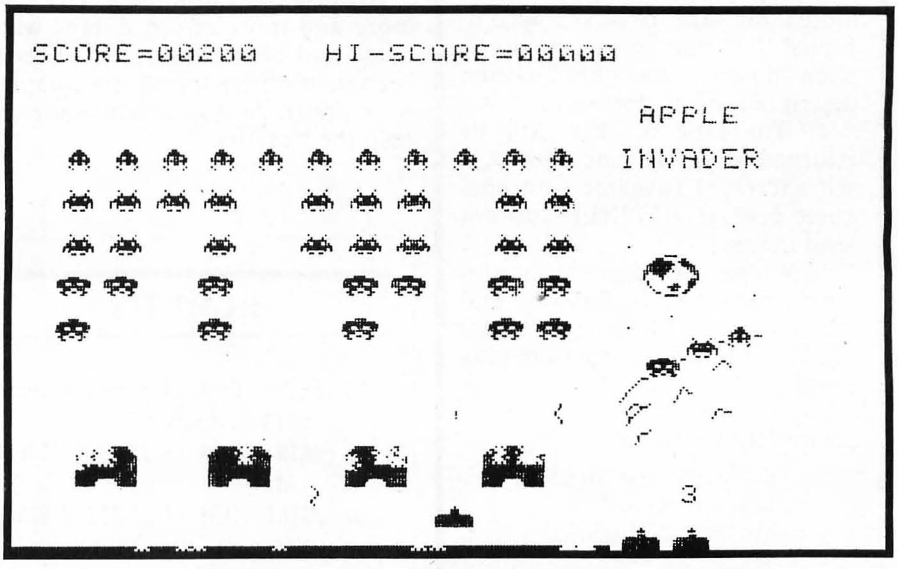
All text functions are software con-



Bronto in the Swamp



Apple Invader (reverse screen dump)



Apple Invader (positive screen dump)

K-Byters

ANOTHER PROGRAMMING CHALLENGE

Some time ago *SoftSide* began inviting its readers to submit "One Liners" — self-contained single-line programs for the TRS-80®, Apple, or ATARI® which would provide a continuously changing graphics display. The response has been excellent, and we're still looking for more submissions.

Now we have a new challenge for you as well: "K-Byters." A K-Byter is a BASIC program which fits into 1K (1024) bytes of program memory. There aren't any restrictions on the nature of the program, other than its size. It can be a graphics display, a game, a mini-adventure, or anything your imagination and programming skills can create.

Note that the program does not have to RUN in 1K of memory; it can use as much RAM for arrays, strings, graphics mapping, etc., as you need. We'd prefer that it be able to run in a 16K system, but this is not an absolute limit.

Here then are the official rules:

1. The program must be written for the Apple, TRS-80®, or ATARI®, entirely in BASIC (although it may create and call Machine Language routines).
2. The program must occupy no more than 1024 bytes of memory before running.
3. The program must be submitted on tape or disk, accompanied by your name, address, phone number, and a brief written description of its operation.
4. The tape or disk will be returned only if accompanied by a self-addressed envelope with adequate postage AFFIXED (do not send money).
5. Winners will have their programs published in *SoftSide* and will receive a $10 software certificate for their programming excellence!

Send submissions to:

K-Byters, c/o *SoftSide*
6 South Street
Milford, NH 03055

trollable. The user can software select any one of nine print fonts by altering pitch and slant. Lines of text are printed bidirectionally or unidirectionally by choice and the user can change line density from six lines per inch to eight lines per inch. The comprehensive user manual states that print-head life exceeds one million characters, however in actual use, an off-the-shelf head has printed five million characters.

DIP printers are designed for 100% duty cycle without overheating. Almost all heat generated inside a printer comes from the printing head. A cooling fan inside DIP printers is located in such a position as to force air directly over the print head, thus keeping temperature to a safe operating level.

If you or your company plan to purchase a DIP, my suggestion is to call Steve Cutter at DIP, Inc. (617/482-4214). Steve is in charge of making DIPs available to the individual who wants a very fine printer at low cost. Sure, you can get a better printer with more features, but many small companies and a great number of individuals cannot afford to spend $2000+ for a printer that will do no more than one of the DIP models.

I recommend the purchase of a soundproofing cover and a 2K buffer with your DIP if you intend to use the printer for graphics printouts. The cover will save your ears from the roar of a graphics dump, and the 2K buffer enables faster printing of graphics information from the computer. And one more thing: Purchase your DIP with the serial cable and buy a parallel one. For some reason the DIP will cost less overall and you never know when you may need the serial cable for a modem hookup or whatever.

Because of *SoftSide*'s DIP 84G, more and more screen dumps will be made and eventually screen shots will become obsolete for all the computers we support. And you, dear reader, will reap the benefits.

HGR2 TO 1

```
10  PRINT  CHR$ (4);"BSAVE SCREEN
     ,A$4000,L$2000"
20  PRINT  CHR$ (4);"BLOAD SCREEN
     ,A$2000"
30  PRINT  CHR$ (4);"DELETE SCREE
     N"
```

LONEG

```
10  PR# 1
20  PRINT  CHR$ (9);"510N"
30  PRINT  CHR$ (27); CHR$ (8); CHR$
     (1)
40  PRINT " ": PRINT " "
50  FOR Y = 0 TO 39
60  PRINT  CHR$ (27); CHR$ (6)
70  FOR I = 1 TO 100: PRINT  CHR$
     (0);: NEXT I
80  FOR X = 0 TO 39
90 C = 63
100  IF  SCRN( X,Y) <  > 0 THEN C
      = 0
110  FOR I = 1 TO 9
120  PRINT  CHR$ (C);
130  NEXT I: NEXT X
140  FOR I = 1 TO 52: PRINT  CHR$
      (0);: NEXT I
150  NEXT Y
160  FOR I = 1 TO 5
170  PRINT " "
180  NEXT I
190  PR# 0
```

LOPOS

```
10  PR# 1
20  PRINT  CHR$ (9);"510N"
30  PRINT  CHR$ (27); CHR$ (8); CHR$
     (1)
40  PRINT " ": PRINT " "
50  FOR Y = 0 TO 39
60  PRINT  CHR$ (27); CHR$ (6)
70  FOR I = 1 TO 100: PRINT  CHR$
     (0);: NEXT I
80  FOR X = 0 TO 39
90 C = 0
100  IF  SCRN( X,Y) <  > 0 THEN C
      = 63
110  FOR I = 1 TO 9
120  PRINT  CHR$ (C);
130  NEXT I: NEXT X
140  FOR I = 1 TO 52: PRINT  CHR$
      (0);: NEXT I
150  NEXT Y
160  FOR I = 1 TO 5
170  PRINT " "
180  NEXT I
190  PR# 0
```

SOLITAIRE

by Larry Williams

Translations by Alan J. Zett and Rich Bouchard.

***Solitaire* is a graphic card game for a 24K Apple or ATARI®, or 16K TRS-80®.**

Solitaire games are among the most popular card games. Although they lack the action of the arcade games, they can be every bit as addictive. What starts out as "one quick game" often turns into an evening of "one more game."

There are many variations of solitaire; however, the most widely known version is the game of Klondike. Both the lay-out and the rules of the game are simple. It is this simplicity that is perhaps its strength.

THE RULES OF KLONDIKE

1. Use one deck of playing cards.
2. Deal twenty-eight cards into seven piles. The first pile on the left contains one card, the second contains two cards, and so forth.
3. Deal the top card of each pile face up.
4. Build on each pile in descending sequence and alternating colors. (Example: Any red ten may be played on any black jack.)
5. You are always entitled to have seven piles. If you clear away a pile leaving a space, you may play any king to the space.
6. The remaining twenty-four cards form the "stock." Cards are turned up one at a time from the stock onto the "waste" pile. The top card of the waste pile is always in play.
7. Any complete column of cards may be picked up as a unit and played on another pile. (Example: A column of cards consisting of the 9 of hearts, 8 of spades, and 7 of diamonds may be played as a unit on a 10 of spades.)
8. When a face-down card of a pile is uncovered, it is brought into play by turning it face up.
9. Whenever an ace comes into play, use it to start a "foundation" pile. Foundation piles are started to the side of the seven original piles of the tableau.
10. You build up on the foundation piles in suit and sequence.
11. The bottom card of any column of cards is eligible for play to its foundation.
12. Once a card is played to its foundation, it is out of play for the rest of the game.
13. The object of the game is to play each card to its foundation pile.

When the game program is RUN you will find the seven piles of the tableau laid out horizontally across the top of the playing surface. The first card in play is displayed in the lower right corner. A cursor is positioned at the bottom of the screen, and a summary of the commands available is displayed. When you start foundation piles they will appear at the right of the screen, above the waste pile.

The following single key commands are available:

LEFT ARROW: Moves the cursor to the left.
RIGHT ARROW: Moves the cursor to the right.
P: Picks up the card or column of cards above the cursor.
D: Drops cards which have been previously picked up at the cursor position, if the play is legal. You may always drop a card back where you picked it up.
N: Brings the next card from the stock into play by placing it on the top of the waste pile in the lower right of the screen. You are allowed only one pass through the stock.
F: Plays the card positioned above the cursor to its foundation pile, if the play is legal. (You need not type P to pick up a card that you wish to play to a foundation. Typing F alone will automatically move the card from its current location to its proper foundation.)
E: When you have no more legal plays, type E to end the game. The program will request confirmation of this command. Typing E will end your current game and begin a new one.

NOTES ON THE APPLE VERSION

The Apple version of *Solitaire* consists of two programs, which must be typed in and SAVEd separately. The game program itself is too large to fit below page 1 of Hi-Res graphics; therefore, a loader program is first RUN to load the program above the graphics screen and to load the shape table below the graphics screen.

The loader program contains an extensive set of data statements which hold the shape table elements. Because proofreading this many data statements can be a pain, the loader program checks the data statements for accuracy. If an error is found program execution is ended and an error message is displayed.

The program counts the number of elements read until it is supposed to read an end-of-line element (either 9999 or 999). If it encounters this element too early, you will get the message "Line no. XXXX is too short." If it does not encounter this element at the correct time, you will get the message "Line no. XXXX is too long." If the end-of-line element is OK, the program compares the total of the elements of the line read with the correct total for that line. If they are not equal, you will get the message "Line no. XXXX is incorrect." If a line passes all three tests, you will get the message "Line no. XXXX is ok," and program execution will continue.

While the loader program is reading the shape table elements from the data statements, it POKEs each shape table value into memory. When this is completed, the program displays each of the shapes on the Hi-Res screen for your inspection.

Once you have verified that the loader program works correctly to this point, be sure to interrupt it and SAVE it before continuing. Since it LOADs the game program, you will not have it available for saving following that. If the shapes appear OK, your data statements are correct, and you will be asked if you are loading from tape or disk. On disk the game program should be called SOLITAIRE.PGM, to correspond to the name in line 810 of the loader program.

Once everything is working properly, you probably will not want to keep the portions of the loader program which check the accuracy of the DATA statements. The following changes can then be made:

```
DEL 20,200
DEL 215,220
DEL 250,280
DEL 300,300
DEL 390,390
DEL 410,410
DEL 420,430
DEL 520,600
DEL 8000,9010
20 VTAB 10: PRINT "LOADING
SHAPES"
200 AD = 7569: CK = 999
240 READ D: IF D = CK THEN 290
```

An even more efficient arrangement can be made if you are storing your program on disk. After confirming that the shapes have been properly POKEd into memory, save the shape table directly to disk by typing BSAVE SOLITAIRE.SHAPES, A7569, L623. The loader program can then be reduced to the following lines:

```
10 HOME: D$=CHR$(4)
20 PRINT D$;"BLOAD
SOLITAIRE.SHAPES"
30 POKE 232,145: POKE 233,29
40 POKE 103,1: POKE 104,64: POKE
16384,0
50 PRINT D$;"RUN SOLITAIRE.
PGM"
```

Variables (Game Program, All Versions)

A$: Temporary string variable.
AZ: Loop variable.
BL,GN,WH,RD,BU: Set to the HCOLOR values for black, green, white, red, and blue.
C: Set equal to VA.
C(6,11): Array of card values in the tableau.
C$: Card values in letter form.
CHR: Related to ASCII value of current card value.
CPOS: Memory location of character data for current card value.
CU: Value of current cursor position (1-7).
D(51): The deck of cards.
D$: Ctrl-D for disk operations.
F(4): Array of top cards on foundation piles.
G$(5): Graphics strings.
HF: Set to 1 if you have picked up cards in your hand, 0 if you do not.
I,IN,J: Counters.
IN(7): Array of counters pointing to the next open element in the tableau array and open deck.
NUM: Card and suit value.
OC: Old cursor position.
OD(23): Array for open deck.
PAUSE: Delay loop variable.
P(6,5): Array of face-down cards in tableau piles.
RN: Set to seed number for RND.
T: Temporary variable.
TX$(13): Array of text messages.
ST: Cursor position at which cards were last picked up.
SU: Suit of the card.
VA: Face value of the card.
VA$: Card values in letter form.
X: Loop variable.
X(7): X-coordinate values for tableau piles, waste pile (open deck), and foundations.
Y1,Y2,Y3,Y4: Y-coordinates for the four foundation piles.
Y(0-12): Y-coordinates for cards played to the tableau.
Y(13): Y-coordinate for the waste pile.
Z,ZA: Loop variables.

Variables (Apple Loader Program)

AD: Memory address to which shape table elements are POKEd.
CK: Equals end-of-line element.
D: Shape table element to be POKEd.
D$: Ctrl-D for disk access.
I,J: Loop counters.
LN: Equals the line number of the data statement being read.
T: Index for array T(0-34).
T(0-34): Array of check-sums.
TT: Accumulates the total of the elements in a data line.
X: X-coordinate of shape.
Y: Y-coordinate of shape.

Apple Loader Program

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       APPLESOFT BASIC        $
$ 'SOLITAIRE LOADER PROGRAM'   $
$   AUTHOR: LARRY WILLIAMS     $
$    (C) 1982     SOFTSIDE     $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialization

```
10  HOME :D$ =  CHR$ (4)
20  DIM T(34):T = 2:LN = 9000:CK =
     9999
```

Read DATA line 8000.

```
30  READ T(0),T(1)
```

Read DATA lines 9000 and 9010 and check their accuracy.

```
40  FOR J = 0 TO 1
42 TT = 0
45  PRINT "LINE NO. ";LN + 10 * J
    ;
50  FOR I = 1 TO 17
60  READ T(T)
70  IF T(T) = CK AND I <  > 17 THEN
      PRINT " IS TOO SHORT.": END

80  IF I = 17 AND TT <  > T(J) THEN
      PRINT " IS INCORRECT.": END

90  IF I = 17 AND T(T) <  > CK THEN
      PRINT " IS TOO LONG.": END

95 TT = TT + T(T): IF I = 17 THEN
     100
98 T = T + 1
100  NEXT
110  PRINT " IS OK."
120  NEXT
```

Read DATA lines 10000-10300, poke the shape table elements into memory, and check the accuracy of each line.

```
200 LN = 10000:CK = 999:AD = 7569

210  FOR J = 0 TO 30
215 TT = 0
220  PRINT "LINE NO. ";LN + 10 *
     J;
230  FOR I = 1 TO 21
240  READ D: IF D = CK THEN 250
245  POKE AD,D:AD = AD + 1
250  IF D = CK AND I <  > 21 THEN
      PRINT " IS TOO SHORT.": END

260  IF I = 21 AND TT <  > T(J +
     2) THEN  PRINT " IS INCORREC
     T.": END
270  IF I = 21 AND D <  > CK THEN
      PRINT " IS TOO LONG.": END

280 TT = TT + D
290  NEXT
300  PRINT " IS OK."
310  NEXT
```

Read and poke DATA line 10310.

```
400  FOR I = 1 TO 3: READ D: POKE
     AD,D
405 AD = AD + 1
415  NEXT
```

Poke starting address of shape table into memory.

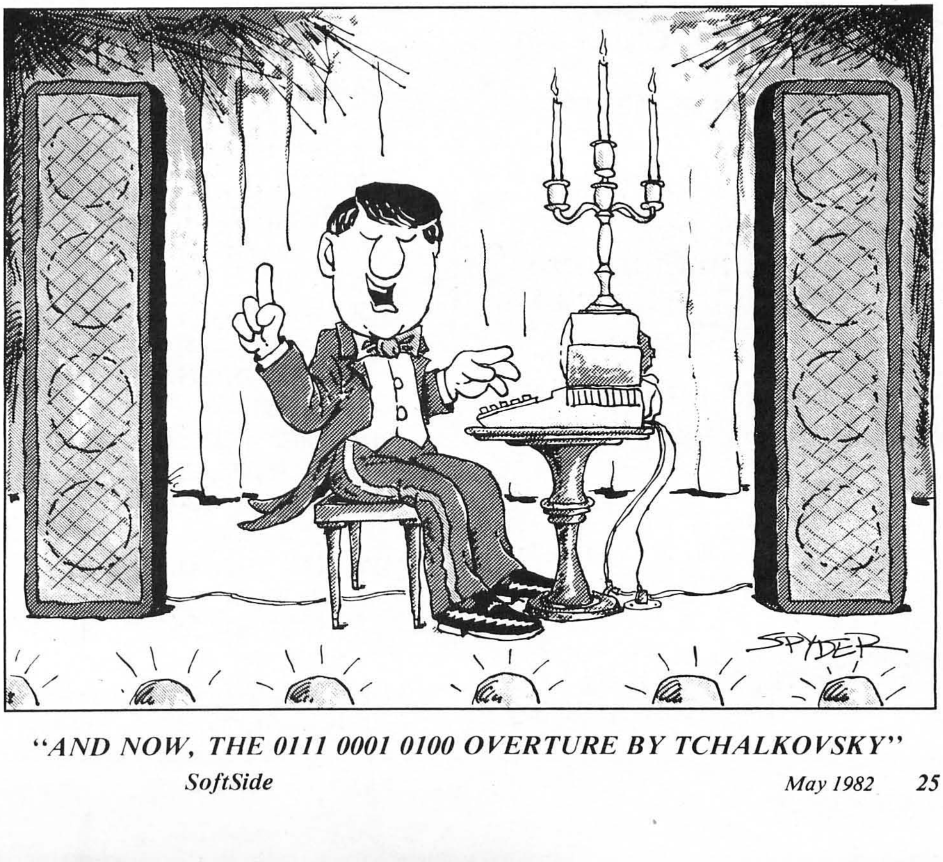
```
500  POKE 232,145: POKE 233,29
```

Draw shapes on the Hi-Res screen for visual verification.

```
520  ROT= 0: SCALE= 1: HCOLOR= 3:
      HGR : HOME
530 X = 10:Y = 20
540  FOR I = 1 TO 18
550  DRAW I AT X,Y:X = X + 20: IF
     X > 259 THEN X = 10:Y = Y +
     20: NEXT
560  NEXT
600  VTAB 23: INPUT "HIT <RETURN>
      TO BEGIN LOADING THE GAME."
     ;A$
```

Load Solitaire program from tape or disk, above the Hi-Res screen in memory.

```
610  HOME : TEXT : VTAB 10: PRINT
     "    1. LOAD FROM TAPE"
620  VTAB 12: PRINT "    2. LOAD
     FROM DISK"
630  VTAB 22: INPUT "CHOOSE ONE:
     ";A$
640  IF  VAL (A$) = 0 OR  VAL (A$
     ) > 2 THEN 610
650  ON  VAL (A$) GOSUB 700,800
700  HOME : VTAB 10: PRINT "CUE T
     HE TAPE TO THE BEGINNING OF
     THE    SOLITAIRE PROGRAM, ST
     ART THE TAPE, THEN PRESS <RE
     TURN>."
710  PRINT : PRINT "AFTER TWO 'BE
     EPS' THE PROMPT SIGN WILL  R
     ETURN.  TURN OFF THE RECORDE
     R AND TYPE 'RUN' TO BEGIN TH
     E GAME."
```



"AND NOW, THE 0111 0001 0100 OVERTURE BY TCHALKOVSKY"

```
720  INPUT "";A$
725  POKE 103,01: POKE 104,64: POKE
     16384,0
730  LOAD
800  HOME : VTAB 10: PRINT "NOW L
     OADING SOLITAIRE PROGRAM"
805  POKE 103,01: POKE 104,64: POKE
     16384,0
810  PRINT D$"RUN SOLITAIRE.PGM"
```

DATA lines.

```
8000  DATA  17930,18473
9000  DATA  811,1478,1114,819,953
     ,1037,1048,715,901,1007,1655
     ,954,1041,1276,1528,1593,999
     9
9010  DATA  1235,1058,1191,1389,1
     306,1210,1282,1085,1039,1202
     ,999,1058,1183,1827,1392,17,
     9999
10000  DATA  18, 0, 38, 0, 50, 0,
      65, 0, 77, 0, 89, 0, 101, 0
     , 114, 0, 123, 0, 136, 0, 99
     9
10010  DATA  148, 0, 164, 0, 173,
      0, 186, 0, 199, 0, 50, 1, 1
     48, 1, 241, 1, 95, 2, 33, 36
     , 999
10020  DATA  100, 12, 14, 14, 54,
      63, 119, 9, 46, 0, 41, 45,
     45, 216, 219, 16, 12, 12, 45
     , 32, 999
10030  DATA  28, 63, 30, 7, 0, 1,
      168, 45, 5, 32, 28, 103, 12
     , 60, 63, 63, 0, 73, 33, 5,
     999
10040  DATA  56, 63, 39, 12, 12,
     12, 54, 46, 0, 1, 112, 45, 5
     , 32, 228, 63, 39, 44, 45, 4
     5, 999
10050  DATA  0, 9, 45, 5, 32, 28,
      63, 214, 36, 100, 12, 45, 5
     , 0, 33, 100, 12, 12, 228, 5
     8, 999
10060  DATA  63, 7, 0, 9, 45, 12,
      228, 63, 214, 36, 32, 12, 4
     5, 14, 54, 0, 41, 101, 12, 6
     0, 999
10070  DATA  63, 7, 32, 12, 45, 2
     1, 46, 0, 33, 36, 36, 108, 9
     , 45, 14, 54, 54, 30, 63, 7,
      999
10080  DATA  32, 36, 36, 0, 1, 11
     2, 45, 5, 32, 36, 36, 4, 0,
     9, 109, 28, 223, 108, 13, 36
     , 999
10090  DATA  228, 95, 191, 54, 7,
      0, 33, 36, 36, 108, 9, 30,
     30, 30, 14, 14, 14, 5, 0, 73
     , 999
10100  DATA  9, 45, 45, 45, 229,
     59, 63, 12, 109, 73, 56, 255
     , 59, 223, 63, 7, 40, 45, 10
     9, 109, 999
10110  DATA  45, 45, 5, 56, 63, 6
     3, 63, 63, 63, 63, 63, 7, 40
     , 45, 45, 45, 45, 45, 45, 45
     , 999
10120  DATA  45, 60, 63, 63, 63,
     63, 63, 63, 63, 63, 44, 45,
     45, 45, 45, 45, 45, 45, 45,
     28, 999
10130  DATA  63, 63, 63, 63, 63,
     63, 63, 39, 45, 45, 45, 45,
     45, 45, 45, 229, 63, 63, 63,
      63, 999
10140  DATA  63, 63, 103, 45, 45,
      45, 45, 45, 229, 63, 63, 63
     , 63, 103, 45, 45, 45, 229,
     63, 63, 999
10150  DATA  103, 45, 229, 39, 45
     , 0, 73, 9, 45, 45, 45, 229,
      59, 63, 12, 109, 73, 56, 25
     5, 59, 999
10160  DATA  223, 63, 7, 40, 45,
     109, 109, 45, 45, 5, 56, 63,
      63, 63, 63, 63, 63, 63, 7,
     40, 999
10170  DATA  45, 45, 45, 45, 45,
     45, 45, 45, 60, 63, 63, 63,
     63, 63, 63, 63, 63, 44, 45,
     45, 999
10180  DATA  45, 45, 45, 45, 45,
     45, 28, 63, 63, 63, 63, 63,
     63, 63, 103, 41, 109, 45, 10
     9, 45, 999
10190  DATA  220, 27, 63, 255, 8,
      45, 45, 45, 56, 63, 63, 63,
      44, 45, 45, 45, 28, 63, 63,
      103, 999
10200  DATA  45, 229, 63, 0, 73,
     73, 9, 37, 255, 40, 45, 45,
     60, 63, 63, 31, 40, 45, 45,
     45, 999
10210  DATA  45, 60, 63, 63, 63,
     63, 31, 200, 45, 45, 45, 45,
      45, 45, 37, 63, 63, 63, 63,
      63, 999
10220  DATA  63, 255, 40, 45, 45,
      45, 45, 45, 45, 45, 45, 60,
      63, 63, 63, 63, 63, 63, 63,
      63, 999
10230  DATA  76, 45, 45, 45, 45,
     45, 45, 37, 63, 63, 63, 63,
     63, 63, 103, 41, 45, 45, 45,
      45, 999
10240  DATA  60, 63, 63, 63, 63,
     76, 45, 45, 45, 24, 63, 63,
     103, 41, 60, 7, 0, 73, 73, 9
     , 999
10250  DATA  37, 255, 40, 45, 45,
      60, 63, 63, 31, 40, 45, 45,
      45, 45, 60, 63, 63, 63, 63,
      31, 999
10260  DATA  40, 45, 45, 45, 45,
     45, 45, 60, 63, 63, 63, 63,
     63, 63, 31, 40, 45, 45, 45,
     45, 999
10270  DATA  45, 45, 45, 45, 60,
     63, 63, 63, 63, 63, 63, 63,
     63, 44, 45, 45, 45, 45, 45,
     45, 999
10280  DATA  45, 45, 60, 63, 63,
     63, 63, 63, 63, 63, 63, 76,
     45, 45, 109, 41, 45, 45, 60,
      63, 999
10290  DATA  63, 223, 63, 63, 39,
      45, 45, 109, 41, 45, 45, 22
     0, 255, 219, 27, 103, 43, 29
     , 77, 73, 999
10300  DATA  201, 37, 255, 219, 2
     7, 63, 0, 45, 45, 45, 45, 28
     , 63, 63, 63, 12, 45, 45, 28
     , 63, 999
10310  DATA    12, 5, 0
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$       APPLESOFT BASIC        $
$    'SOLITAIRE PROGRAM'       $
$   AUTHOR: LARRY WILLIAMS     $
$   (C) 1982    SOFTSIDE       $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


10  GOTO 1110
```

Subroutine to draw cards.

```
20  HCOLOR= WH: FOR J = 0 TO 29: HPLOT
     X,Y + J TO X + 29,Y + J: NEXT

30  HCOLOR= BL: DRAW VA AT X + 1,
     Y + 7: DRAW VA AT X + 20,Y +
     27: DRAW SU + 13 AT X + 6,Y +
     22
40  HCOLOR= BL: DRAW VA AT X + 2,
     Y + 7: DRAW VA AT X + 21,Y +
     27: DRAW SU + 13 AT X + 7,Y +
     22
50  IF SU > 2 THEN  HCOLOR= RD: DRAW
     SU + 13 AT X + 6,Y + 22
60  RETURN
```

Subroutine to get SU and VA.

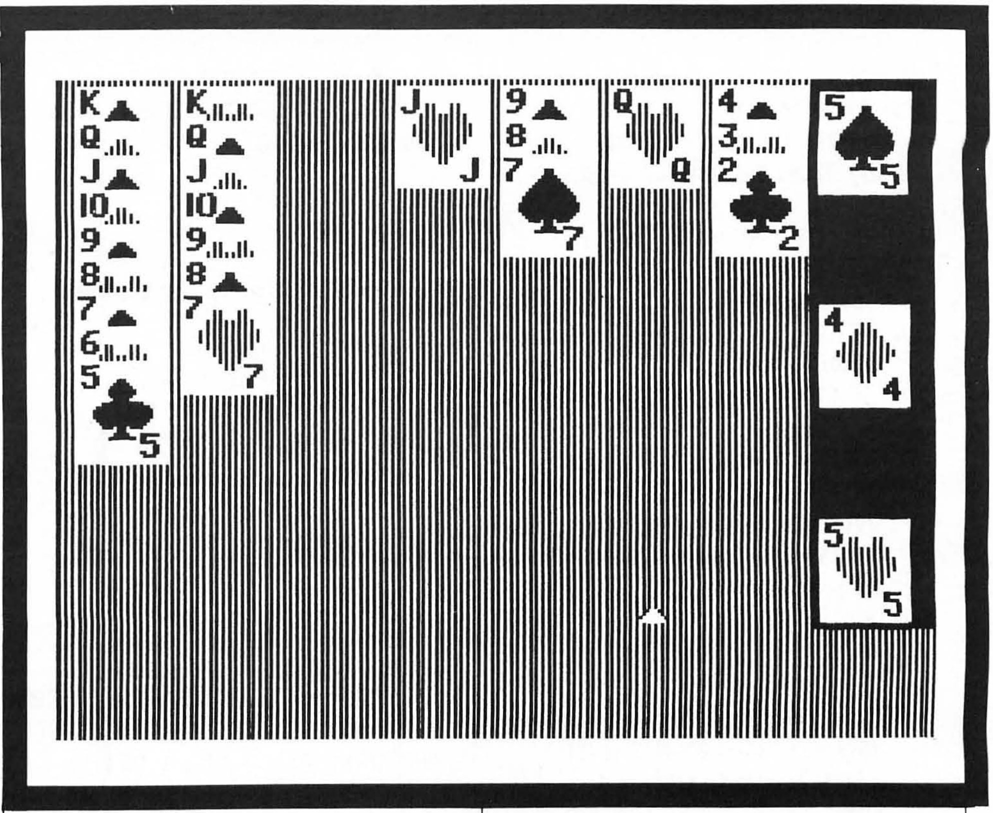
```
70 SU =  INT (NUM / 100)
80 VA = NUM - 100 * SU
90  RETURN
```

Subroutine for next card.

```
100  IF HF THEN  GOSUB 1480: RETURN

110  IF IN > 51 THEN 1490: RETURN

120 OD(IN(7)) = D(IN):IN = IN + 1
     :X = X(7):Y = Y(13):NUM = OD
     (IN(7)): GOSUB 70: GOSUB 20:
     IN(7) = IN(7) + 1
130  IF CU = 7 THEN  GOSUB 150
140  RETURN
```

Subroutine to draw cursor.

```
150  HCOLOR= BU: DRAW 18 AT X(OC)
      + 11,157: DRAW 18 AT X(OC) +
     10,157: DRAW 18 AT X(OC) + 9
     ,157
160  IF OC = 7 THEN  HCOLOR= WH: DRAW
     18 AT X(OC) + 11,157: DRAW 1
     8 AT X(OC) + 10,157: DRAW 18
      AT X(OC) + 9,157
165  IF (OC = 7) * (IN(7) = 0) THEN
      HCOLOR= BL: DRAW 18 AT X(OC
```



```
     ) + 10,157: DRAW 18 AT X(OC)
      + 9,157: DRAW 18 AT X(OC) +
     11,157
170  HCOLOR= BL: DRAW 18 AT X(CU)
      + 11,157: DRAW 18 AT X(CU) +
     10,157: DRAW 18 AT X(CU) + 9
     ,157
180  HCOLOR= WH: DRAW 18 AT X(CU)
      + 10,157
190  RETURN
```

Subroutine to move right.

```
200 CU = CU + 1: IF CU > 7 THEN C
     U = 7
210  GOSUB 150
220 OC = CU
230  RETURN
```

Subroutine to move left.

```
240 CU = CU - 1: IF CU < 0 THEN C
     U = 0
250  GOSUB 150
260 OC = CU
270  RETURN
```

Subroutine to pick up cards.

```
280  IF HF THEN  GOSUB 1480: RETURN

290 ST = CU
300  IF IN(CU) = 0 THEN  GOSUB 15
     10: RETURN
310  IF CU = 7 THEN NUM = OD(IN(7
     ) - 1): GOTO 330
320 NUM = C(CU,0)
330 HF = 1
340 J = 0: IF CU = 7 THEN  HCOLOR=
     WH:J = 13: GOTO 360
350  HCOLOR= BU: IF P(CU,0) <  >
     0 THEN  HCOLOR= GN
360  FOR I = 0 TO 29: HPLOT X(CU)
     ,Y(J) + I TO X(CU) + 29,Y(J)
      + I
370  NEXT
380  IF CU = 7 THEN  GOSUB 150: RETURN

385  IF IN(CU) = 1 THEN  RETURN
390 J = 10 * (IN(CU) - 1) + 29
395  HCOLOR= BU: FOR I = 30 TO J:
      HPLOT X(CU),Y(0) + I TO X(C
     U) + 29,Y(0) + I: NEXT
400  RETURN
```

Main subroutine to drop card.

```
410  IF  NOT HF THEN  GOSUB 1520:
      RETURN
420  IF CU = 7 THEN  GOSUB 590: RETURN

430  IF ST = CU THEN  GOSUB 750: RETURN
440  IF IN(CU) = 0 THEN  GOSUB 63
     0: RETURN
450 NUM = C(CU,IN(CU) - 1)
460  GOSUB 70:TS = SU:TV = VA
470  IF ST = 7 THEN NUM = OD(IN(7
     ) - 1): GOTO 490
```

```
480 NUM = C(ST,0)
490  GOSUB 70: IF ((TS = 1) + (TS
      = 2)) * ((SU = 1) + (SU = 2
     )) THEN  GOSUB 1530: RETURN

500  IF ((TS = 3) + (TS = 4)) * (
     (SU = 3) + (SU = 4)) THEN  GOSUB
     1540: RETURN
510  IF TV <  > VA + 1 THEN  GOSUB
     1550: RETURN
520  IF ST = 7 THEN  GOSUB 700: RETURN

530  FOR I = 0 TO IN(ST) - 1:NUM =
     C(ST,I):C(CU,IN(CU)) = NUM: GOSUB
     70:X = X(CU):Y = Y(IN(CU)): GOSUB
     20:IN(CU) = IN(CU) + 1:C(ST,
     I) = 0: NEXT
540 IN(ST) = 0:HF = 0
550  IF P(ST,0) = 0 THEN  RETURN

560 NUM = P(ST,0): GOSUB 70:X = X
     (ST):Y = Y(0): GOSUB 20:C(ST
     ,IN(ST)) = NUM:IN(ST) = 1
570  FOR I = 0 TO 4:P(ST,I) = P(S
     T,I + 1): NEXT :P(ST,5) = 0
580  RETURN
```

Drop card back on open deck.

```
590  IF ST <  > 7 THEN  GOSUB 156
     0: RETURN
600 NUM = OD(IN(7) - 1): GOSUB 70
     :X = X(CU):Y = Y(13): GOSUB
     20: GOSUB 150
610 HF = 0
620  RETURN
```

Drop king on open space.

```
630  IF ST = 7 THEN NUM = OD(IN(7
     ) - 1): GOTO 650
640 NUM = C(ST,0)
650  GOSUB 70
660  IF VA <  > 13 THEN  GOSUB 15
     70: RETURN
670  IF ST = 7 THEN  GOSUB 700: RETURN

680  GOSUB 530
690  RETURN
```

Drop card from open deck.

```
700 X = X(CU):Y = Y(IN(CU)):C(CU,
     IN(CU)) = NUM: GOSUB 20:IN(C
     U) = IN(CU) + 1
710 IN(7) = IN(7) - 1:OD(IN(7)) =
     0:HF = 0
720  IF IN(7) = 0 THEN  HCOLOR= B
     L:X = X(7):Y = Y(13): FOR J =
     0 TO 29: HPLOT X,Y + J TO X +
     29,Y + J: NEXT : RETURN
730 NUM = OD(IN(7) - 1): GOSUB 70
     :X = X(7):Y = Y(13): GOSUB 2
     0
740  RETURN
```

Drop cards back where you got them.

```
750  FOR I = 0 TO IN(CU) - 1:NUM =
     C(CU,I): GOSUB 70:X = X(CU):
     Y = Y(I): GOSUB 20: NEXT
760 HF = 0
770  RETURN
```

Play to foundations from open deck.

```
780 NUM = OD(IN(7) - 1): GOSUB 70
     :FL = 1
785  IF (F(SU) <  > VA - 1) * (F(
     SU) = 0) THEN  GOSUB 1580: RETURN

790  IF F(SU) <  > VA - 1 THEN TV
      = F(SU): GOSUB 1550: RETURN

800  GOSUB 980
810 OD(IN(CU)) = 0
820  IF IN(CU) = 0 THEN  GOSUB 72
     0: RETURN
830  GOSUB 730
835  GOSUB 150
840  RETURN
```

Play last card of column to foundation.

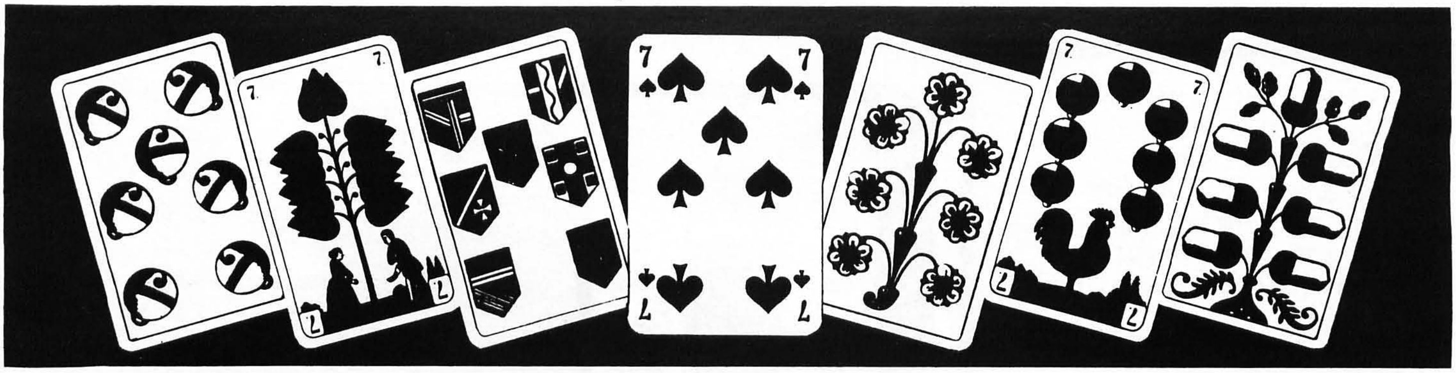
```
850  HCOLOR= BU
860  IF P(CU,0) <  > 0 THEN  HCOLOR=
     GN
870  FOR I = 0 TO 29: HPLOT X(CU)
     ,Y(0) + I TO X(CU) + 29,Y(0)
      + I: NEXT :C(CU,0) = P(CU,0
     )
880  IF P(CU,0) = 0 THEN  RETURN

890 NUM = C(CU,0):X = X(CU):Y = Y
     (0): GOSUB 70: GOSUB 20
900 IN(CU) = 1
910  FOR I = 0 TO 4:P(CU,I) = P(C
     U,I + 1): NEXT :P(CU,5) = 0
920  RETURN
```

Main subroutine to play to foundations.

```
930  IF HF THEN  RETURN
935 FL = 0
940  IF IN(CU) = 0 THEN  GOSUB 15
     10: RETURN
950  IF CU = 7 THEN  GOSUB 780: RETURN
960 NUM = C(CU,IN(CU) - 1): GOSUB
     70
965  IF (F(SU) <  > VA - 1) * (F(
     SU) = 0) THEN  GOSUB 1580: RETURN

970  IF F(SU) <  > VA - 1 THEN TV
      = F(SU): GOSUB 1550: RETURN

980 X = X(7)
990  IF SU = 1 THEN Y = Y1
1000  IF SU = 2 THEN Y = Y2
1010  IF SU = 3 THEN Y = Y3
1020  IF SU = 4 THEN Y = Y4
1030  GOSUB 20:F(SU) = VA
1040 IN(CU) = IN(CU) - 1: IF FL THEN
      RETURN
1050 C(CU,IN(CU)) = 0
1060  IF IN(CU) = 0 THEN  GOSUB 8
     50: RETURN
1070 X = X(CU):Y = Y(IN(CU) - 1):
     NUM = C(CU,IN(CU) - 1): GOSUB
     70: GOSUB 20
1080  HCOLOR= BU
1085  IF IN(CU) + 2 = 13 THEN  FOR
     I = 0 TO 9: HPLOT X(CU),132 +
     I TO X(CU) + 29,132 + I: NEXT
     : RETURN
1090  FOR I = 0 TO 9: HPLOT X(CU)
     ,Y(IN(CU) + 2) + I TO X(CU) +
     29,Y(IN(CU) + 2) + I: NEXT
1100  RETURN
```

Initialize variables.

```
1110  DIM C(6,11),P(6,5),D(51),OD
     (23),F(4),X(7),Y(13),IN(7),T
     X$(13)
1120  FOR I = 1 TO 13: READ TX$(I
     ): NEXT
1130 BL = 4:GN = 1:WH = 7:RD = 5:
     BU = 6
1140 D$ =  CHR$ (4)
1150  FOR I = 0 TO 6: FOR J = 0 TO
     5:C(I,J) = 0:P(I,J) = 0: NEXT
     : FOR J = 6 TO 11:C(I,J) = 0
     : NEXT : NEXT
1160  FOR I = 0 TO 23:OD(I) = 0: NEXT

1170  FOR I = 0 TO 3:F(I) = 0: NEXT

1180  FOR I = 0 TO 7:X(I) = 4 + I
     * 34:Y(I) = 2 + I * 10: NEXT

1190  FOR I = 8 TO 12:Y(I) = 2 +
     I * 10: NEXT
1200 Y1 = 4:Y2 = 35:Y3 = 66:Y4 =
     97:Y(13) = 128
```



```
1210 IN = 0: FOR I = 1 TO 4: FOR
     J = 1 TO 13:D(IN) = 100 * I +
     J:IN = IN + 1: NEXT : NEXT
```

Set up random function

```
1220 RN =  PEEK (78) + 256 *  PEEK
     (79):X =  RND ( - RN)
```

Shuffle cards.

```
1230  FOR I = 51 TO 0 STEP  - 1:X
      =  INT ( RND (1) * I + 1):T
      = D(X):D(X) = D(I):D(I) = T
     : NEXT
```

Deal to tableau.

```
1240 IN = 0: FOR I = 1 TO 6: FOR
     J = 0 TO I - 1:P(I,J) = D(IN
     ):IN = IN + 1: NEXT : NEXT
1250  FOR I = 0 TO 6:C(I,0) = D(I
     N):IN = IN + 1: NEXT
```

Draw set-up.

```
1260  HGR : SCALE= 1: ROT= 0: POKE
     34,20
1270  HOME : HCOLOR= BU: HPLOT 0,
     0: CALL 62454
1280  HCOLOR= BL: FOR I = 240 TO
     279: HPLOT I,0 TO I,159: NEXT

1290  FOR I = 0 TO 6:NUM = C(I,0)
     : GOSUB 70:X = X(I):Y = Y(0)
     : GOSUB 20: NEXT I
1300  FOR I = 0 TO 6:IN(I) = 1: NEXT
     :IN(7) = 0
1310  GOSUB 100: GOSUB 1460
1320 CU = 0:OC = 0:X = X(CU): GOSUB
     150
```

Control program.

```
1330  VTAB 1: GET A$
1340  IF  ASC (A$) = 21 THEN  GOSUB
     200: GOTO 1330
1350  IF  ASC (A$) = 8 THEN  GOSUB
     240: GOTO 1330
1360  IF A$ = "N" THEN  GOSUB 100
     : GOTO 1330
1370  IF A$ = "P" THEN  GOSUB 280
     : GOTO 1330
1380  IF A$ = "D" THEN  GOSUB 410
     : GOTO 1330
1390  IF A$ <  > "F" THEN 1400
1392  GOSUB 930: IF F(1) < 13 OR
     F(2) < 13 OR F(3) < 13 OR F(
     4) < 13 THEN 1330
1394  HOME : VTAB 22: INPUT "YOU
     WIN!  DO YOU WANT TO PLAY AG
     AIN? ";A$:A$ =  LEFT$ (A$,1)

1396  TEXT : HOME : IF A$ <  > "N
     " THEN  RUN
1398  END
1400  IF A$ = "E" THEN  GOTO 1420

1410  GOTO 1330
```

Text messages.

```
1420  HOME : VTAB 22: INPUT "DO Y
     OU WANT TO END THE GAME? (Y/
     N) ";A$
1430  IF A$ <  > "Y" THEN  GOSUB
     1460: GOTO 1330
1440  TEXT : HOME : RUN
1450  VTAB 1: END
1460  HOME : PRINT "--> = MOVE RI
     GHT    D = DROP CARDS      <
     -- = MOVE LEFT      N = NEXT
     CARD          P  = PICK UP CAR
     DS E = END GAME
        F = PLAY TO FOUNDATIONS";

1470  RETURN
1480  HOME : VTAB 22: PRINT "YOU
     ALREADY HAVE A HANDFUL OF CA
     RDS.    YOU MUST 'DROP' THEM
      BEFORE YOU MAKE ANOTHER PLA
     Y.";: FOR PAUSE = 0 TO 3000:
      NEXT : GOSUB 1460: RETURN
1490  HOME :: VTAB 22: PRINT "THE
     RE ARE NO MORE CARDS IN THE
     DECK.    YOU MUST CONTINUE P
     LAY WITH THE CARDS   SHOWING
     .";: FOR PAUSE = 0 TO 3000: NEXT
     : GOSUB 1460
1500  RETURN
1510  HOME : VTAB 22: PRINT "THER
     E ARE NO CARDS HERE TO PICK
     UP.": FOR PA = 0 TO 3000: NEXT
     : GOSUB 1460: RETURN
1520  HOME : VTAB 22: PRINT "YOU
     DO NOT HAVE ANY CARDS TO DRO
     P.";: FOR PA = 0 TO 3000: NEXT
     : GOSUB 1460: RETURN
1530  HOME : VTAB 22: PRINT "YOU
     CAN'T PLAY A BLACK CARD ON A
      BLACK  CARD.": FOR PA = 0 TO
     3000: NEXT : GOSUB 1460: RETURN

1540  HOME : VTAB 22: PRINT "YOU
     CAN'T PLAY A RED CARD ON A R
     ED CARD": FOR PA = 0 TO 3000
     : NEXT : GOSUB 1460: RETURN

1550  HOME : VTAB 22: PRINT "YOU
     CAN'T PLAY A"TX$(VA): PRINT
     "ON A"TX$(TV): FOR PA = 0 TO
     3000: NEXT : GOSUB 1460: RETURN

1560  HOME : VTAB 22: PRINT "YOU
     CAN'T DROP THEM HERE.": FOR
     PA = 0 TO 3000: NEXT : GOSUB
     1460: RETURN
1570  HOME : VTAB 22: PRINT "YOU
     CAN ONLY DROP A KING ON AN E
     MPTY    SPACE.": FOR PA = 0 TO
     3000: NEXT : GOSUB 1460: RETURN

1580  HOME : VTAB 22: PRINT "YOU
     MUST START YOUR FOUNDATION W
     ITH AN  ACE.": FOR PA = 0 TO
     3000: NEXT : GOSUB 1460: RETURN
```

Names of cards.

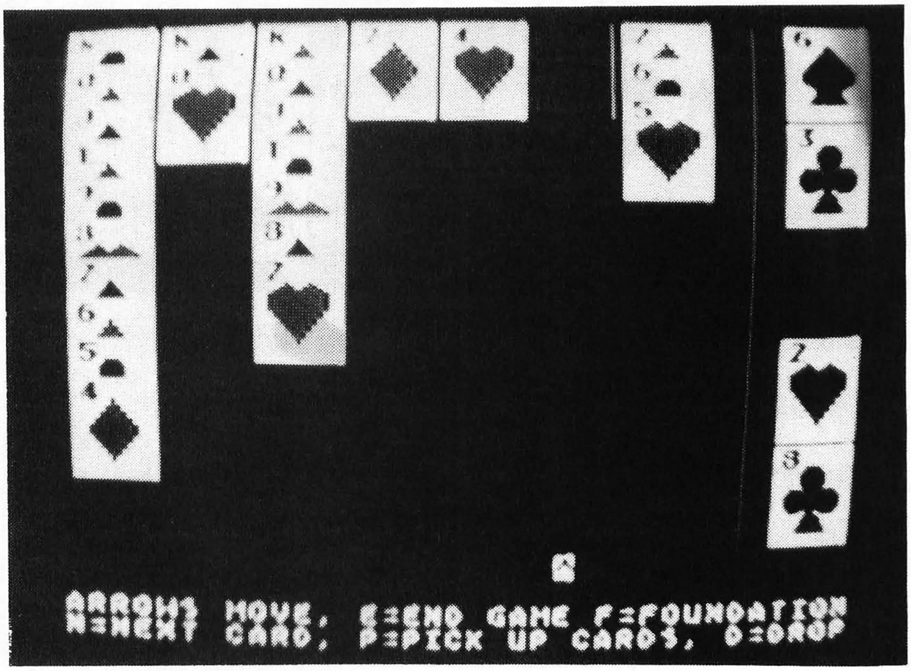
```
1590  DATA   "N ACE"," TWO"," THR
     EE"," FOUR"," FIVE"," SIX","
      SEVEN","N EIGHT", " NINE",
     " TEN"," JACK"," QUEEN"," KI
     NG"
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$         ATARI BASIC          $
$         "SOLITAIRE"          $
$   AUTHOR: LARRY WILLIAMS     $
$     TRANSL: ALAN J. ZETT     $
$      (C) 1982, SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
1 GRAPHICS 18:POKE 756,209
10 POSITION 5,3:? #6;"sOlItAiRe":POSIT
ION 3,5:? #6;"TRANSLATED BY":POSITION
4,7:? #6;"alan j zett":GOTO 1110
```

Subroutine to draw cards.

```
20 COLOR 1:FOR AZ=Y TO Y+30:PLOT X,AZ:
DRAWTO X+30,AZ:NEXT AZ
22 C=VA:ON SU GOTO 36,46,24,28
23 REM DIAMONDS
24 COLOR 0:FOR ZA=0 TO 9:FOR AZ=0 TO Z
A STEP 2:PLOT X+15+AZ,Y+ZA+6:PLOT X+15
-AZ,Y+ZA+6:NEXT AZ:NEXT ZA
26 FOR ZA=9 TO 0 STEP -1:FOR AZ=0 TO Z
A STEP 2:PLOT X+15+AZ,Y-ZA+25:PLOT X+1
5-AZ,Y-ZA+25:NEXT AZ:NEXT ZA:GOTO 56
27 REM HEARTS
28 COLOR 0:FOR ZA=0 TO 10:FOR AZ=0 TO
ZA STEP 2:PLOT X+15+AZ,Y-ZA+25:PLOT X+
15-AZ,Y-ZA+25:NEXT AZ:NEXT ZA
30 FOR ZA=11 TO 15:FOR AZ=0 TO 10 STEP
 2:PLOT X+15+AZ,Y-ZA+25:PLOT X+15-AZ,Y
-ZA+25:NEXT AZ:NEXT ZA
32 Z=-2:FOR ZA=16 TO 18:Z=Z+2:FOR AZ=Z
 TO 8-Z STEP 2:PLOT X+15+AZ,Y-ZA+25:PL
OT X+15-AZ,Y-ZA+25:NEXT AZ:NEXT ZA
34 GOTO 56
35 REM SPADES
36 COLOR 0:FOR AZ=0 TO 9:PLOT X+15+AZ,
Y+AZ+6:DRAWTO X+15-AZ,Y+AZ+6:NEXT AZ
38 FOR AZ=10 TO 12:PLOT X+15+9,Y+AZ+6:
DRAWTO X+15-9,Y+AZ+6:NEXT AZ
40 Z=-2:FOR ZA=13 TO 15:Z=Z+2:FOR AZ=Z
 TO 8-Z:PLOT X+15+AZ,Y+ZA+6:PLOT X+15-
AZ,Y+ZA+6:NEXT AZ:NEXT ZA
42 Z=0:FOR AZ=13 TO 19:Z=Z+1:PLOT X+15
+Z,Y+AZ+6:DRAWTO X+15-Z,Y+AZ+6:NEXT AZ
44 GOTO 56
45 REM CLUBS
46 COLOR 0:AZ=X+11:ZA=Y+6:GOSUB 52:AZ=
X+5:ZA=Y+13:GOSUB 52:AZ=X+17:GOSUB 52
48 Z=0:FOR AZ=14 TO 21:Z=Z+0.65:PLOT X
+15+Z,Y+AZ+5:DRAWTO X+15-Z,Y+AZ+5:NEXT
 AZ
50 FOR Z=X+14 TO X+16:PLOT Z,Y+13:DRAW
TO Z,Y+19:NEXT Z:GOTO 56
52 PLOT AZ+2,ZA:DRAWTO AZ+6,ZA:PLOT AZ
+2,ZA+7:DRAWTO AZ+6,ZA+7:PLOT AZ+1,ZA+
1:DRAWTO AZ+7,ZA+1
```



```
ARROWS MOVE; E=END GAME F=FOUNDATION
N=NEXT CARD, P=PICK UP CARDS, D=DROP
```

```
54 PLOT AZ+1,ZA+6:DRAWTO AZ+7,ZA+6:FOR
 Z=ZA+2 TO ZA+5:PLOT AZ,Z:DRAWTO AZ+8,
Z:NEXT Z:RETURN
55 REM CHR PRINT
56 CHR=ASC(C$(C,C)):IF CHR<96 THEN CHR
=CHR-32*(CHR>31)+64*(CHR<32)
58 CPOS=CHR*8+(PEEK(756)*256)
60 FOR AZ=0 TO 7:POKE (AZ*40)+MEM+(X/8
)+(Y*40),255-PEEK(CPOS+AZ):NEXT AZ:RET
URN
```

Subroutine to get SU and VA.

```
70 SU=INT(NUM/100)
80 VA=NUM-100*SU
90 RETURN
```

Subroutine for next card.

```
100 IF HF THEN GOSUB 1480:RETURN
110 IF IN>51 THEN 1490
120 OD(IN(7))=D(IN):IN=IN+1:X=X(7):Y=Y
(13):NUM=OD(IN(7)):GOSUB 70:GOSUB 20:I
N(7)=IN(7)+1
140 RETURN
```

Subroutine to draw cursor.

```
150 POKE 656,0:IF OC<7 THEN POKE 657,4
*OC+4:? " "
160 IF OC=7 THEN POKE 657,35:? " "
170 POKE 656,0:IF CU<7 THEN POKE 657,4
*CU+4:? "^"
180 IF CU=7 THEN POKE 657,35:? "^"
```

```
190 RETURN
```

Subroutine to move right.

```
200 CU=CU+1:IF CU>7 THEN CU=7
210 GOSUB 150
220 OC=CU
230 RETURN
```

Subroutine to move left.

```
240 CU=CU-1:IF CU<0 THEN CU=0
250 GOSUB 150
260 OC=CU
270 RETURN
```

Subroutine to pick up cards.

```
280 IF HF THEN GOSUB 1480:RETURN
290 ST=CU
300 IF IN(CU)=0 THEN GOSUB 1510:RETURN
310 IF CU=7 THEN NUM=OD(IN(7)-1):GOTO
330
320 NUM=C(CU,0)
330 HF=1
340 J=0:IF CU=7 THEN COLOR 1:FOR I=0 T
O 30:PLOT X(CU)+I,Y(13):DRAWTO X(CU)+I
,Y(13)+30:GOTO 370
350 IF P(CU,0)=0 THEN COLOR 0:FOR I=0
TO 30:PLOT X(CU)+I,Y(J):DRAWTO X(CU)+I
,Y(J)+30:GOTO 370
360 FOR I=0 TO 30 STEP 2:COLOR 0:PLOT
X(CU)+I,Y(J):DRAWTO X(CU)+I,Y(J)+30
365 IF I<30 THEN COLOR 1:PLOT X(CU)+I+
1,Y(J):DRAWTO X(CU)+I+1,Y(J)+30
 370 NEXT I
 380 IF CU=7 THEN GOSUB 150:RETURN
 390 J=12*(IN(CU)-1)+32
395 COLOR 0:FOR I=0 TO 30:PLOT X(CU)+I
,Y(0)+31:DRAWTO X(CU)+I,Y(0)+J:NEXT I
 400 RETURN
```

Main subroutine to drop card.

```
 410 IF  NOT HF THEN GOSUB 1520:RETURN
 420 IF CU=7 THEN GOSUB 590:RETURN
430 IF ST=CU THEN GOSUB 750:RETURN
440 IF IN(CU)=0 THEN GOSUB 630:RETURN
450 NUM=C(CU,IN(CU)-1)
 460 GOSUB 70:TS=SU:TV=VA
470 IF ST=7 THEN NUM=OD(IN(7)-1):GOTO
 490
480 NUM=C(ST,0)
490 GOSUB 70:IF ((TS=1) OR (TS=2)) AND
 ((SU=1) OR (SU=2)) THEN GOSUB 1530:RE
TURN
500 IF ((TS=3) OR (TS=4)) AND ((SU=3)
OR (SU=4)) THEN GOSUB 1540:RETURN
 510 IF TV<>VA+1 THEN GOSUB 1550:RETURN
 520 IF ST=7 THEN GOSUB 700:RETURN
530 FOR I=0 TO IN(ST)-1:NUM=C(ST,I):C(
CU,IN(CU))=NUM:GOSUB 70:X=X(CU):Y=Y(IN
(CU)):GOSUB 20:IN(CU)=IN(CU)+1
 540 C(ST,I)=0:NEXT I:IN(ST)=0:HF=0
550 IF P(ST,0)=0 THEN RETURN
560 NUM=P(ST,0):GOSUB 70:X=X(ST):Y=Y(0
):GOSUB 20:C(ST,IN(ST))=NUM:IN(ST)=1
570 FOR I=0 TO 4:P(ST,I)=P(ST,I+1):NEX
T I:P(ST,5)=0
580 RETURN
```

Drop card back on open deck.

```
590 IF ST<>7 THEN GOSUB 1560:RETURN
600 NUM=OD(IN(7)-1):GOSUB 70:X=X(CU):Y
=Y(13):GOSUB 20:GOSUB 150
 610 HF=0
620 RETURN
```

Drop king on open space.

```
630 IF ST=7 THEN NUM=OD(IN(7)-1):GOTO
650
640 NUM=C(ST,0)
650 GOSUB 70
660 IF VA<>13 THEN GOSUB 1570:RETURN
670 IF ST=7 THEN GOSUB 700:RETURN
680 GOSUB 530
690 RETURN
```

Drop card from open deck.

```
700 X=X(CU):Y=Y(IN(CU)):C(CU,IN(CU))=N
UM:GOSUB 20:IN(CU)=IN(CU)+1
710 IN(7)=IN(7)-1:OD(IN(7))=0:HF=0
720 IF IN(7)=0 THEN COLOR 0:X=X(7):Y=Y
(13):FOR I=0 TO 30:PLOT X+I,Y:DRAWTO X
+I,Y+30:NEXT I:RETURN
730 NUM=OD(IN(7)-1):GOSUB 70:X=X(7):Y=
Y(13):GOSUB 20
740 RETURN
```

Drop cards back where you got them.

```
750 FOR I=0 TO IN(CU)-1:NUM=C(CU,I):GO
SUB 70:X=X(CU):Y=Y(I):GOSUB 20:NEXT I
760 HF=0
770 RETURN
```

Play to foundations from open deck.

```
780 NUM=OD(IN(7)-1):GOSUB 70:FL=1
785 IF (F(SU)<>VA-1) AND (F(SU)=0) THE
N GOSUB 1580:RETURN
 790 IF F(SU)<>VA-1 THEN TV=F(SU):GOSUB
 1550:RETURN
 800 GOSUB 980
810 OD(IN(CU))=0
820 IF IN(CU)=0 THEN GOSUB 720:RETURN
830 GOSUB 730
835 GOSUB 150
840 RETURN
```

Play last card of column to foundation.

```
850 IF P(CU,0)=0 THEN COLOR 0:FOR I=0
TO 30:PLOT X(CU)+I,Y(0):DRAWTO X(CU)+I
,Y(0)+30:GOTO 875
860 FOR I=0 TO 30 STEP 2:COLOR 0:PLOT
X(CU)+I,Y(0):DRAWTO X(CU)+I,Y(0)+30
870 IF I<30 THEN COLOR 1:PLOT X(CU)+I+
1,Y(0):DRAWTO X(CU)+I+1,Y(0)+30
 875 NEXT I:C(CU,0)=P(CU,0)
880 IF P(CU,0)=0 THEN RETURN
890 NUM=C(CU,0):X=X(CU):Y=Y(0):GOSUB 7
0:GOSUB 20
900 IN(CU)=1
910 FOR I=0 TO 4:P(CU,I)=P(CU,I+1):NEX
T I:P(CU,5)=0
920 RETURN
```

Main subroutine to play to foundations.

```
930 IF HF THEN RETURN
935 FL=0
940 IF IN(CU)=0 THEN GOSUB 1510:RETURN
950 IF CU=7 THEN GOSUB 780:RETURN
960 NUM=C(CU,IN(CU)-1):GOSUB 70
965 IF (F(SU)<>VA-1) AND (F(SU)=0) THE
N GOSUB 1580:RETURN
 970 IF F(SU)<>VA-1 THEN TV=F(SU):GOSUB
 1550:RETURN
980 X=X(7)
990 IF SU=1 THEN Y=Y1
1000 IF SU=2 THEN Y=Y2
1010 IF SU=3 THEN Y=Y3
1020 IF SU=4 THEN Y=Y4
1030 GOSUB 20:F(SU)=VA
1040 IN(CU)=IN(CU)-1:IF FL THEN RETURN
1050 C(CU,IN(CU))=0
1060 IF IN(CU)=0 THEN GOSUB 850:RETURN
1070 X=X(CU):Y=Y(IN(CU)-1):NUM=C(CU,IN
(CU)-1):GOSUB 70:GOSUB 20
1080 COLOR 0
1090 FOR I=31 TO 45:PLOT X(CU),Y(IN(CU
)-1)+I:DRAWTO X(CU)+30,Y(IN(CU)-1)+I:N
EXT I
1100 RETURN
```

Initialize variables.

```
1110 CLR :DIM TX$(91),AZ$(7),C$(13),C(
6,11),P(6,5),D(51),OD(23),F(4),X(7),Y(
13),IN(7):OPEN #1,4,0,"K"
 1115 TX$(91)="#":TX$(1,1)=" ":TX$(2)=T
X$(1):C$="A23456789TJQK":AZ$=TX$
 1120 FOR I=1 TO 13:AZ$="       ":READ
AZ$:TX$(I*7-6,I*7)=AZ$:NEXT I
1150 FOR I=0 TO 6:FOR J=0 TO 5:C(I,J)=
0:P(I,J)=0:NEXT J:FOR J=6 TO 11:C(I,J)
=0:NEXT J:NEXT I
1160 FOR I=0 TO 23:OD(I)=0:NEXT I
1170 FOR I=0 TO 4:F(I)=0:NEXT I
1180 FOR I=0 TO 7:X(I)=21+I*32:Y(I)=I*
12:NEXT I:X(7)=269
1190 FOR I=8 TO 12:Y(I)=I*12:NEXT I
1200 Y1=0:Y2=32:Y3=64:Y4=96:Y(13)=128
1210 POKE 756,224
1220 IN=0:FOR I=1 TO 4:FOR J=1 TO 13:D
(IN)=100*I+J:IN=IN+1:NEXT J:NEXT I
```

Shuffle cards.

```
1230 FOR I=51 TO 0 STEP -1:X=INT(RND(0
)*I+1):T=D(X):D(X)=D(I):D(I)=T:NEXT I
```

Deal to tableau.

```
1240 IN=0:FOR I=1 TO 6:FOR J=0 TO I-1:
P(I,J)=D(IN):IN=IN+1:NEXT J:NEXT I
1250 FOR I=0 TO 6:C(I,0)=D(IN):IN=IN+1
:NEXT I
```

Draw set-up.

```
1260 GRAPHICS 8:POKE 710,96:POKE 752,1
:COLOR 1:POKE 709,15:POKE 712,4:MEM=PE
EK(88)+PEEK(89)*256
```

```
1280 FOR X=254 TO 258 STEP 2:PLOT X,0:
DRAWTO X,160:NEXT X
1290 FOR I=0 TO 6:NUM=C(I,0):GOSUB 70:
X=X(I):Y=Y(0):GOSUB 20:NEXT I
1300 FOR I=0 TO 6:IN(I)=1:NEXT I:IN(7)
=0
1310 GOSUB 100:GOSUB 1460
1320 CU=0:OC=0:X=X(CU):GOSUB 150
```

Control program.

```
1330 POKE 764,255:GET #1,A:A=A-32*(A>9
0)
1340 IF A=42 THEN GOSUB 200:GOTO 1330
1350 IF A=43 THEN GOSUB 240:GOTO 1330
1360 IF A=78 THEN GOSUB 100:GOTO 1330
1370 IF A=80 THEN GOSUB 280:GOTO 1330
1380 IF A=68 THEN GOSUB 410:GOTO 1330
1390 IF A=70 THEN GOSUB 930:GOTO 1620
1400 IF A=69 THEN 1420
1410 GOTO 1330
```

Text messages.

```
1420 GOSUB 1600:? "DO YOU WANT TO END
THE GAME? (Y/N)";:GET #1,A
1430 IF A<>89 THEN GOSUB 1460:GOTO 133
0
1440 RUN
1450 GOSUB 1600:END
1460 GOSUB 1600:? "ARROWS MOVE, E=END
GAME F=FOUNDATION  N=NEXT CARD, P=PICK
 UP CARDS, D=DROP";
1470 RETURN
1480 GOSUB 1600:? "YOU'VE ALREADY PICK
ED UP A CARD";:GOTO 1610
1490 GOSUB 1600:? "THERE ARE NO MORE C
ARDS IN THE DECK   YOU MUST PLAY WITH
THE CARDS SHOWING";:GOTO 1610
1510 GOSUB 1600:? "THERE ARE NO CARDS
HERE TO PICK UP";:GOTO 1610
1520 GOSUB 1600:? "YOU DO NOT HAVE ANY
 CARDS TO DROP";:GOTO 1610
1530 GOSUB 1600:? "YOU CAN'T PLAY BLAC
K ON BLACK";:GOTO 1610
1540 GOSUB 1600:? "YOU CAN'T PLAY RED
ON RED";:GOTO 1610
1550 GOSUB 1600:? "YOU CAN'T DROP A";T
X$(VA*7-6,VA*7):? "ON TOP OF A";TX$(TV
*7-6,TV*7);:GOTO 1610
1560 GOSUB 1600:? "YOU CAN'T DROP CARD
S HERE";:GOTO 1610
1570 GOSUB 1600:? "YOU CAN ONLY DROP A
 KING HERE";:GOTO 1610
1580 GOSUB 1600:? "START YOUR FOUNDATI
ON WITH AN ACE";:GOTO 1610
```

Names of cards.

```
1590 DATA N ACE, TWO, THREE, FOUR, FIV
E, SIX, SEVEN,N EIGHT, NINE, TEN, JACK
, QUEEN, KING
```

Routine to clear bottom of screen.

```
1600 POKE 656,2:POKE 657,2:? CHR$(156)
;CHR$(156);CHR$(253);:RETURN
```

Pause subroutine.

```
1610 FOR PAUSE=1 TO 300:NEXT PAUSE:GOS
UB 1460:RETURN
```

Check for game won.

```
1620 IF F(1)<13 OR F(2)<13 OR F(3)<13
OR F(4)<13 THEN GOTO 1330
1630 GOSUB 1600:? " YOU WIN!!  CARE TO
 PLAY AGAIN? (Y/N)";:GET #1,A:IF A<>78
 THEN RUN
1640 GRAPHICS 0:CLR
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$         TRS-80 BASIC         $
$          "SOLITAIRE"         $
$  AUTHOR: LARRY WILLIAMS      $
$  TRANSL: RICH BOUCHARD       $
$      (C) 1982  SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
10 CLS:CLEAR1000:DEFINTA-Z:GOTO 1110
```

Subroutine to draw cards.

```
20 PRINT@X*8+Y*64,G$(6);
30 PRINT@X*8+Y*64,MID$(VA$,VA,1);
50 PRINTG$(SU);
60 RETURN
```

Subroutine to get SU and VA.

```
70 SU=INT(NU/100)
80 VA=NU-100*SU
90 RETURN
```

Subroutine for next card.

```
100 IF HF<>0 THEN GOSUB1480:RETURN
110 IF IN>51 THEN 1490
120 OD(IN(7))=D(IN):IN=IN+1:X=7:Y=12:NU=OD(IN(7)):GOSUB70:GOSUB2
0:IN(7)=IN(7)+1
130 IF CU=7 THEN GOSUB 150
140 RETURN
```

Subroutine to draw cursor.

```
150 IF OC<>7 THEN PRINT@898+OC*8,"   "; ELSE IF IN(7)<>0 THEN PR
INT@952,CHR$(191); ELSE PRINT@952," ";
160 IF CU<>7 THEN PRINT@898+CU*8,G$(5); ELSE PRINT@952,CHR$(Q1);
190 RETURN
```

Subroutine to move right.

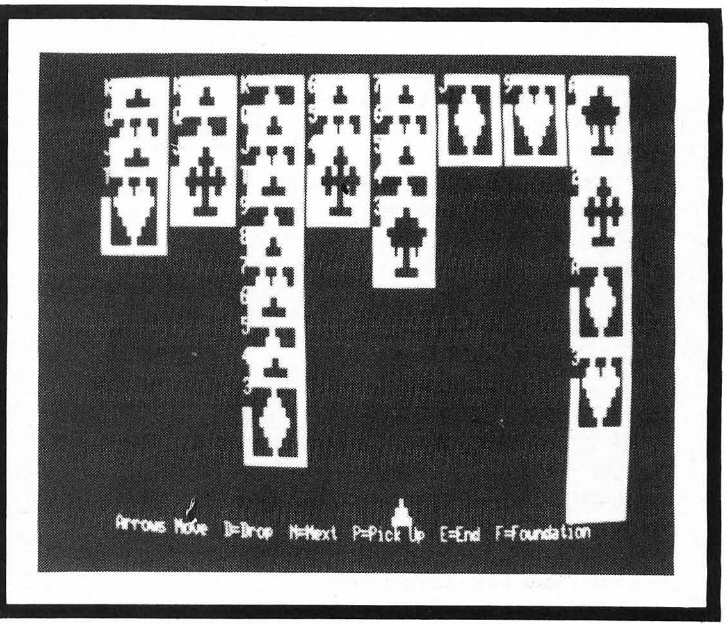
```
200 CU=CU+1:IF CU>7 THEN CU=7
210 GOSUB 150
220 OC=CU
230 RETURN
```

Subroutine to move left.

```
240 CU=CU-1:IF CU<0 THEN CU=0
250 GOSUB 150
260 OC=CU
270 RETURN
```

Subroutine to pick up cards.

```
280 IF HF<>0 THEN GOSUB 1480:RETURN
290 ST=CU
300 IF IN(CU)=0 THEN GOSUB 1510:RETURN
310 IF CU=7 THEN NU=OD(IN(7)-1):GOTO 330
320 NU=C(CU,0)
330 HF=1
```



Arrows MoVe D=Drop N=Next P=Pick Up E=End F=Foundation

```
340 X=CU:J=0:IF CU=7 THEN J=12:PRINT@X*8+768,G$(6);:GOTO 380
350 IF P(CU,0)<>0 THEN PRINT@X*8,G$(6);:GOTO 380
360 FORY=0TO2:PRINT@Y*64+X*8,STRING$(8,32);
370 NEXTY
380 IF CU=7 THEN GOSUB 150:RETURN
385 IF IN(CU)=1 THEN RETURN
390 J=IN(CU)+2:IF J>15 THEN J=15
395 FOR I=3 TO J:PRINT@I*64+X*8,STRING$(8,32);:NEXT I
400 RETURN
```

Main subroutine to drop card.

```
410 IF HF=0 THEN GOSUB 1520:RETURN
420 IF CU=7 THEN GOSUB 590:RETURN
430 IF ST=CU THEN GOSUB 750:RETURN
440 IF IN(CU)=0 THEN GOSUB 630:RETURN
450 NU=C(CU,IN(CU)-1)
460 GOSUB70:TS=SU:TV=VA
470 IF ST=7 THEN NU=OD(IN(7)-1):GOTO 490
480 NU=C(ST,0)
490 GOSUB70:IF((TS=1)OR(TS=2))AND((SU=1)OR(SU=2))THENGOSUB1530:R
ETURN
500 IF((TS=3)OR(TS=4))AND((SU=3)OR(SU=4))THENGOSUB1540:RETURN
510 IF TV<>VA+1 THEN GOSUB1550:RETURN
515 IF VA=1 THEN GOSUB 1500:RETURN
520 IF ST=7 THEN GOSUB700:RETURN
530 FOR I=0 TO IN(ST)-1:NU=C(ST,I):C(CU,IN(CU))=NU:GOSUB70:X=CU:
Y=IN(CU):GOSUB20:IN(CU)=IN(CU)+1:C(ST,I)=0:NEXT I
540 IN(ST)=0:HF=0
550 IF P(ST,0)=0 THEN RETURN
560 NU=P(ST,0):GOSUB70:X=ST:Y=0:GOSUB20:C(ST,IN(ST))=NU:IN(ST)=1
570 FORI=0TO4:P(ST,I)=P(ST,I+1):NEXTI:P(ST,5)=0
580 RETURN
```

Drop card back on open deck.

```
590 IF ST<>7 THEN GOSUB 1560:RETURN
600 NU=OD(IN(7)-1):GOSUB70:X=CU:Y=12:GOSUB20:GOSUB150
610 HF=0
620 RETURN
```

Drop king on open space.

```
630 IF ST=7 THEN NU=OD(IN(7)-1):GOTO 650
640 NU=C(ST,0)
650 GOSUB 70
660 IF VA<>13 THEN GOSUB 1570:RETURN
670 IF ST=7 THEN GOSUB 700:RETURN
680 GOSUB 530
690 RETURN
```

Drop card from open deck.

```
700 X=CU:Y=IN(CU):C(CU,IN(CU))=NU:GOSUB20:IN(CU)=IN(CU)+1
710 IN(7)=IN(7)-1:OD(IN(7))=0:HF=0
720 IF IN(7)=0 THEN FOR Y=12 TO 14:PRINT@Y*64+56,CHR$(30);:NEXT
Y:RETURN
730 NU=OD(IN(7)-1):GOSUB70:X=7:Y=12:GOSUB20
740 RETURN
```

Drop cards back where you got them.

```
750 FOR I=0 TO IN(CU)-1:NU=C(CU,I):GOSUB70:X=CU:Y=I:GOSUB20:NEXT
I
760 HF=0
770 RETURN
```

Play to foundations from open deck.

```
780 NU=OD(IN(7)-1):GOSUB70:FL=1
785 IF (F(SU)<>VA-1) AND (F(SU)=0) THEN GOSUB 1580:RETURN
790 IF F(SU)<>VA-1 THEN TV=F(SU):GOSUB1550:RETURN
800 GOSUB980
810 OD(IN(CU))=0
820 IF IN(CU)=0 THEN GOSUB 720:RETURN
830 GOSUB730
835 GOSUB150
840 RETURN
```

Play last card of column to foundation.

```
850 '
860 X=CU:IF P(CU,0)<>0 THEN PRINT@X*8,G$(6);:GOTO 880
870 FOR Y=0 TO 2:PRINT@Y*64+X*8,STRING$(8,32);:NEXT Y
880 C(CU,0)=P(CU,0):IF P(CU,0)=0 THEN RETURN
890 NU=C(CU,0):X=CU:Y=0:GOSUB70:GOSUB20
900 IN(CU)=1
910 FORI=0TO4:P(CU,I)=P(CU,I+1):NEXTI:P(CU,5)=0
920 RETURN
```

Main subroutine to play to foundations.

```
930 IF HF<>0 THEN GOSUB 1480:RETURN
935 FL=0
940 IF IN(CU)=0 THEN GOSUB 1510:RETURN
950 IF CU=7 THEN GOSUB 780:RETURN
960 NU=C(CU,IN(CU)-1):GOSUB70
965 IF (F(SU)<>VA-1) AND (F(SU)=0) THEN GOSUB 1580:RETURN
970 IF F(SU)<>VA-1 THEN TV=F(SU):GOSUB1550:RETURN
980 X=7
990 Y=SU*3-3
1030 GOSUB20:F(SU)=VA
1040 IN(CU)=IN(CU)-1:IF FL<>0 THEN RETURN
1050 C(CU,IN(CU))=0
1060 IF IN(CU)=0 THEN GOSUB 850:RETURN
1070 X=CU:Y=IN(CU)-1:NU=C(CU,IN(CU)-1):GOSUB70:GOSUB20
1085 PRINT@CU*8+(IN(CU)+2)*64,STRING$(8,32);
1100 RETURN
```

Initialize variables.

```
1110 DIM C(6,11),P(6,5),D(51),OD(23),F(4),IN(7),TX$(13),G$(10)
1112 PRINT@342,CHR$(23);"Solitaire"
1114 GOSUB 2000
1116 IF PEEK(293)=73 THEN Q1=94 ELSE Q1=91
1120 FORI=1TO13:READTX$(I):NEXTI
1150 FORI=0TO6:FORJ=0TO5:C(I,J)=0:P(I,J)=0:NEXTJ:FORJ=6TO11:C(I,
J)=0:NEXTJ,I
1160 FORI=0TO23:OD(I)=0:NEXTI
1170 FORI=0TO3:F(I)=0:NEXTI
1200 '
1210 IN=0:FORI=1TO4:FORJ=1TO13:D(IN)=100*I+J:IN=IN+1:NEXTJ,I
```

Shuffle cards.

```
1230 FORI=51TO0STEP-1:X=INT(RND(0)*I+1):T=D(X):D(X)=D(I):D(I)=T:
NEXTI
```

Deal to tableau.

```
1240 IN=0:FORI=1TO6:FORJ=0TOI-1:P(I,J)=D(IN):IN=IN+1:NEXTJ,I
1250 FORI=0TO6:C(I,0)=D(IN):IN=IN+1:NEXTI
```

Draw set-up.

```
1260 CLS
1290 FORI=0TO6:NU=C(I,0):GOSUB70:X=I:Y=0:GOSUB20:NEXTI
1300 FORI=0TO6:IN(I)=1:NEXTI:IN(7)=0
1310 GOSUB100:GOSUB1460
1320 CU=0:OC=0:X=CU:GOSUB150
```

Control program.

```
1330 A$=INKEY$:IFA$=""THEN1330
1335 IF A$>=CHR$(96) AND A$<=CHR$(127) THEN A$=CHR$(ASC(A$)-32)
1340 IF A$=CHR$(9) THEN GOSUB200:GOTO1330
1350 IF A$=CHR$(8) THEN GOSUB240:GOTO1330
1360 IF A$="N" THEN GOSUB100:GOTO 1330
1370 IF A$="P" THEN GOSUB280:GOTO 1330
1380 IF A$="D" THEN GOSUB 410:GOTO 1330
1390 IF A$<>"F" THEN 1400 ELSE GOSUB 930: IF F(1)<13 OR F(2)<13
OR F(3)<13 OR F(4)<13 THEN 1330 ELSE PRINT@960,CHR$(30);"YOU WIN
!!!  Care to play again? (Y/N)";
1395 PRINTCHR$(14);:A$=INKEY$:PRINTCHR$(15);:IFA$=""THEN1395ELSE
IFA$<>"N"ANDA$<>CHR$(110)THENRUNELSECLEAR50:CLS:END
1400 IF A$="E" THEN 1420
1410 GOTO 1330
```

Text messages.

```
1420 PRINT@960,CHR$(30);"Do you want to end the game? (Y/N) ";
1430 PRINTCHR$(14);:A$=INKEY$:PRINTCHR$(15);:IFA$=""THEN1430
1432 IF A$<>"Y" AND A$<>CHR$(121) THEN GOSUB 1460:GOTO 1330
1440 RUN
1459 STOP
1460 PRINT@960,CHR$(30);"Arrows Move  D=Drop  N=Next  P=Pick Up
 E=End  F=Foundation";
1470 RETURN
1480 A$="You already have card(s) in your hand.  'DROP' them.":G
OSUB2100:RETURN
```

The playing cards illustrated throughout *"Solitaire"* are examples of international designs.

```
1490 A$="There are no more cards in the deck.":GOSUB2100:RETURN
1500 A$="Why don't you play the ace on a foundation!":GOSUB2100:
RETURN
1510 A$="There are no cards here to pick up.":GOSUB2100:RETURN
1520 A$="You do not have any cards to drop.":GOSUB2100:RETURN
1530 A$="You can't play a black card on a black card.":GOSUB2100
:RETURN
1540 A$="You can't play a red card on a red card.":GOSUB2100:RET
URN
1550 A$="You can't play a"+TX$(VA)+" on a"+TX$(TV)+".":GOSUB2100
:RETURN
1560 A$="You can't drop them here.":GOSUB2100:RETURN
1570 A$="You can only drop a King on an empty space.":GOSUB2100:
RETURN
1580 A$="You must start your foundation with an ace.":GOSUB2100:
RETURN
```

Names of cards.

```
1590 DATA "n Ace"," Two"," Three"," Four"," Five"," Six"," Seven
","n Eight"," Nine"," Ten"," Jack"," Queen"," King"
```

Initialization of graphic strings.

```
2000 G$=CHR$(26)+STRING$(6,8)
2010 G$(1)=CHR$(191)+CHR$(159)+CHR$(135)+CHR$(139)+CHR$(175)+CHR
$(191)+G$+CHR$(189)+CHR$(144)+CHR$(144)+CHR$(160)+CHR$(160)+CHR$
(190)+G$+CHR$(191)+CHR$(183)+CHR$(177)+CHR$(178)+CHR$(187)+CHR$(
191)
2020 G$(2)=CHR$(191)+CHR$(191)+CHR$(135)+CHR$(139)+CHR$(191)+CHR
$(191)+G$+CHR$(183)+CHR$(128)+CHR$(145)+CHR$(162)+CHR$(128)+CHR$
(187)+G$+CHR$(191)+CHR$(183)+CHR$(177)+CHR$(178)+CHR$(187)+CHR$(
191)
2030 G$(3)=CHR$(131)+CHR$(131)+CHR$(187)+CHR$(147)+CHR$(131)+CHR
$(171)+G$+" "+CHR$(174)+CHR$(191)+CHR$(191)+CHR$(132)+CHR$(170)+
G$+CHR$(176)+CHR$(176)+CHR$(187)+CHR$(177)+CHR$(176)+CHR$(186)
2040 G$(4)=CHR$(131)+CHR$(187)+CHR$(147)+CHR$(187)+CHR$(147)+CHR
$(171)+G$+CHR$(130)+CHR$(175)+CHR$(191)+CHR$(191)+CHR$(135)+CHR$
(170)+G$+CHR$(176)+CHR$(176)+CHR$(187)+CHR$(177)+CHR$(176)+CHR$(
186)
2050 G$(5)=CHR$(184)+CHR$(189)+CHR$(144)
2060 G$=G$+STRING$(2,8):G$(6)=STRING$(7,191)+CHR$(149)+G$+STRING
$(7,191)+CHR$(149)+G$+STRING$(7,191)+CHR$(149)
2070 VA$="A23456789TJQK"
2080 RETURN
```

General purpose delay subroutine.

```
2100 PRINT@960,CHR$(30);A$;
2110 PRINT@1022,CHR$(143);:FORT=1TO5:IF INKEY$="" THEN NEXTT:PRI
NT@1022," ";:FOR T=1 TO 5:IF INKEY$="" THEN NEXT T:GOTO 2110
2120 GOSUB1460:RETURN
```

A feast of computing ideas.

If you work with a 6502/6809-based system, you're probably hungry for the facts and ideas that will help you understand the inner workings of your computer. You want to go beyond canned software—use your computer for more than games—learn the advanced programming techniques that enable you to get the most out of your 6502/6809 system.

MICRO, The 6502/6809 Journal, gives you page after page, month after month, of solid information to sink your teeth into. **MICRO** is the premier how-to magazine for serious users of the Apple, PET/CBM, OSI, Atari, AIM, SYM, KIM, and all 6809 based systems including the TRS-80 Color Computer. It's a resource journal internationally respected by professionals in business, industry, and education. Every issue of **MICRO** keeps you informed with up-to-the-minute data on new products and publications:

- **hardware catalog** with organized, concise description
- **software catalog** in an easy-to-use format
- **new publications** listed and annotated
- **reviews and evaluations** of significant products

And there's much more:

- **In-depth hardware tutorials** bring expert advice into your home or office.
- **Detailed discussions of programming languages** deepen and broaden your programming ability.
- **Complete program listings** enable you to increase your machine's capabilities.
- **Bibliography of 6502/6809 information** helps you to find pertinent articles in a timely manner.
- **Special monthly features** with in-depth treatment of one subject or system increase your knowledge of the field.
- **Balanced mix of machine-specific and general articles** for your everyday use as well as long-range reference needs.
- **Informative advertising** focused specifically on 6502/6809 machines keeps you abreast of latest developments.
- **Reader feedback** puts you in touch with other micro-computerists.

MICRO is the magazine you need to get the most from your own 6502/6809 system!

To order, send your check or international money order (payable to MICRO) and the order form at left, to:

Subscription Fulfillment
MICRO, Dept. MI
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

Or, for your convenience, call our toll-free number:

1-800-227-1617, Ext. 564
(In California, 800-772-3545, Ext. 564)

and charge your subscription to your MasterCard or VISA. (All orders must be prepaid in U.S. dollars or charged to your MasterCard or VISA.)

SUBSCRIPTION RATES (U.S. dollars)
Yearly subscription (ISSN 027-9002) saves 20% off the single-issue price.

U.S. $24*
Canada $27
Europe $27 ($42 by air mail)
Mexico, Central America, Mideast, North and Central Africa $27 ($48 air)
South America, Far East, South Africa, Australasia $27 ($72 air)

* SPECIAL OFFER—U.S. ONLY: Save even more—30% off single-issue price: 2 years, $42

Dept. S S

NO. 41 $2.50 OCTOBER 1981

MICRO

THE 6502/6809 JOURNAL

Special PET/VIC Feature
A Look at the TRS-80 Color Computer
Step Up to Programmable Motion
A to D Converters for Your Computer

You'll love every byte.

YES! I want to get more from my microcomputer. Please send me

___ year(s) of MICRO at $______/year.
(Outside U.S. and Canada, please indicate via □ surface or □ air mail.)

Name ______________________

Company ______________________

Street ______________________

City ________ State ________ Zip Code ________

□ Check enclosed for $ ________
□ Charge my credit card account
□ VISA □ MasterCard

Signature ______________________

Card number ______________________

Expiration date ______________________

NORMAL LOADING PROCEDURES FOR SOFTSIDE DISKS AND CASSETTES

Follow these procedures unless otherwise instructed by the documentation in the magazine. Back issues may differ in some details.

APPLE

Disks are in 13-sector format, created under DOS 3.2.1. If your system is set up for 16-sector disks (DOS 3.3), first boot your BASICS disk or BRUN BOOT13 from the System Master Diskette, and then insert the ***SoftSide*** disk. A cover/menu program will run automatically.

Tapes LOAD in the normal way. Advance the tape to the beginning of the lead-in tone; stop the tape; insert the plug into the EAR jack; type LOAD; start the tape; and press RETURN. Side two of the tape is a duplicate of side one, unless one or more Integer BASIC programs are included, in which case side two contains the Integer programs.

ATARI®

Disks do not contain DOS.SYS files, and are therefore not bootable by themselves. First boot a disk which contains any version of DOS, then insert the ***SoftSide*** disk and RUN "D:COVER".

Tapes CLOAD in the normal way. If you have difficulty, try this procedure:
(1) Type POKE 54018,54 and press RETURN.
(2) Turn up the volume on your TV.
(3) Type CLOAD and press RETURN once.
(4) Press the PLAY button and listen.
(5) When you hear a steady lead-in tone, press RETURN again.
Side two of the tape is a duplicate of side one.

TRS-80®

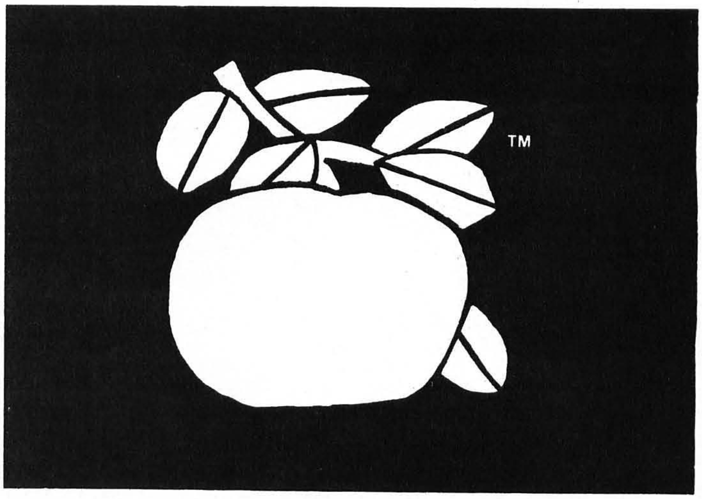
Disks are available in Model I or Model III format. They contain the DOS PLUS operating system, and a cover program which automatically runs upon booting. Back issues prior to May, 1982, are available only in Model I format, and may be converted using the TRSDOS CONVERT utility on a two-drive Model III. Older back issues (with Model I TRSDOS) require you to enter BASIC and then type RUN "COVER".

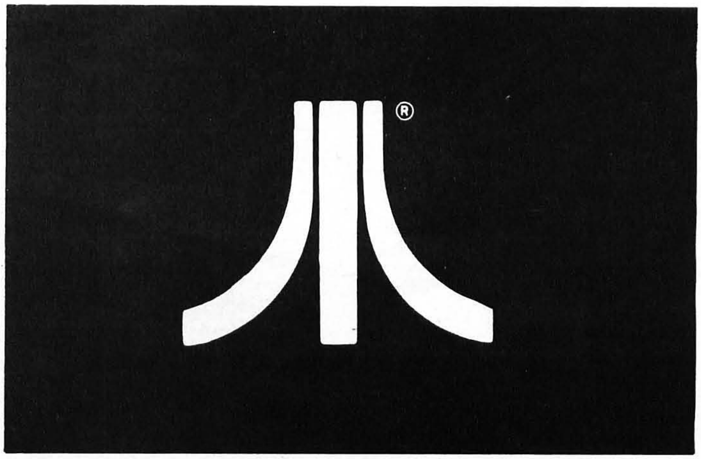
Tapes CLOAD in the normal way on Model I's, and at low speed (500 baud) on Model III's. The first program is a cover/menu program. Side two of the tape is a duplicate of side one.

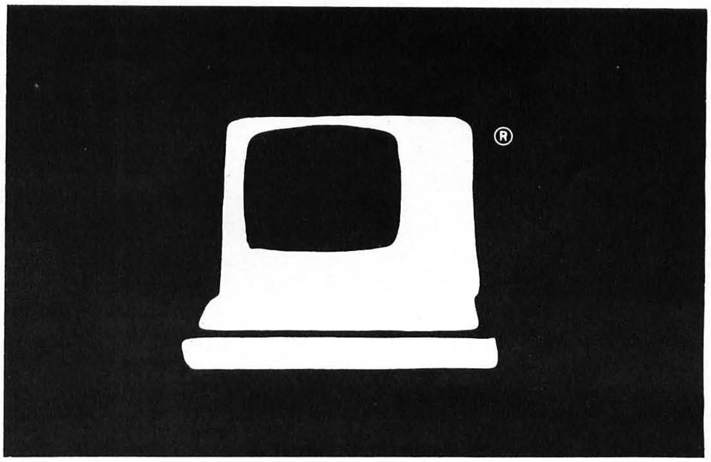
GENERAL NOTES ABOUT MAGNETIC MEDIA

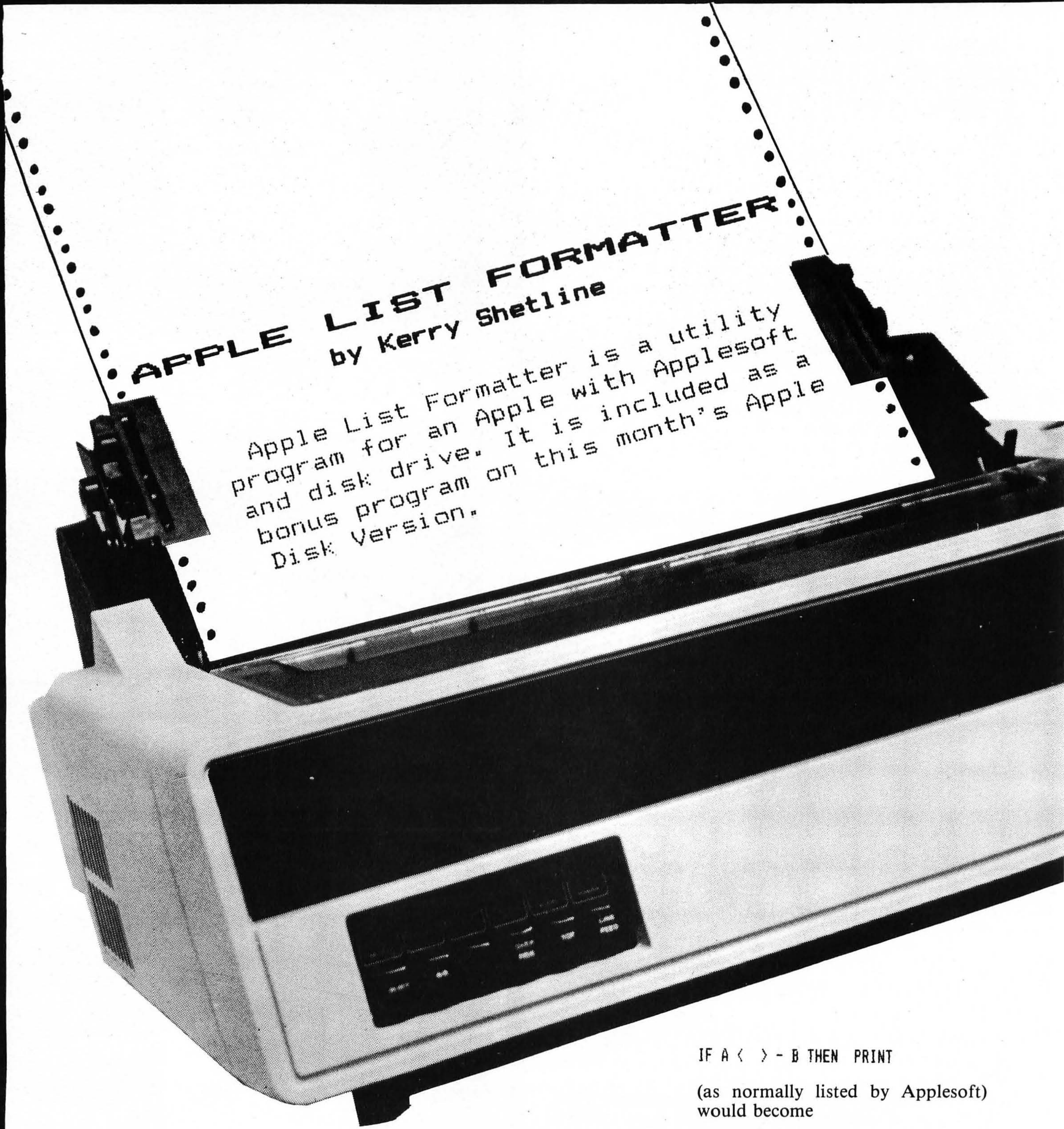
Our disks and tapes are duplicated by reliable, professional duplication services; bad copies are now very rare. However, the trip through the mail does occasionally wreak havoc with sensitive magnetic media. If, after a reasonable number of tries and a careful check and cleaning of your equipment, you are not able to load a program from a tape or disk, please return it to us with an exact description of the problem. If we cannot duplicate the problem on our systems, we will advise you of that when we send the replacement.

We use no copy-protection schemes on our media. We urge you to make a backup copy of every disk or tape as soon as you receive it (and at the same time resist the urge to give copies to friends). Our replacement policy does not extend beyond 30 days.









APPLE LIST FORMATTER

by Kerry Shetline

Apple List Formatter is a utility program for an Apple with Applesoft and disk drive. It is included as a bonus program on this month's Apple Disk Version.

The *List Formatter* is a program designed to list Applesoft programs in a way that is easy to read, pleasing to the eye, and helpful for debugging. The program consists of three parts: an EXEC file, a BASIC "front end," and a Machine Language lister.

The listings produced are different from normal listings in several ways. One major change is in the spacing. Spaces are placed before or after keywords only where it is necessary for clarity. For example, the statement

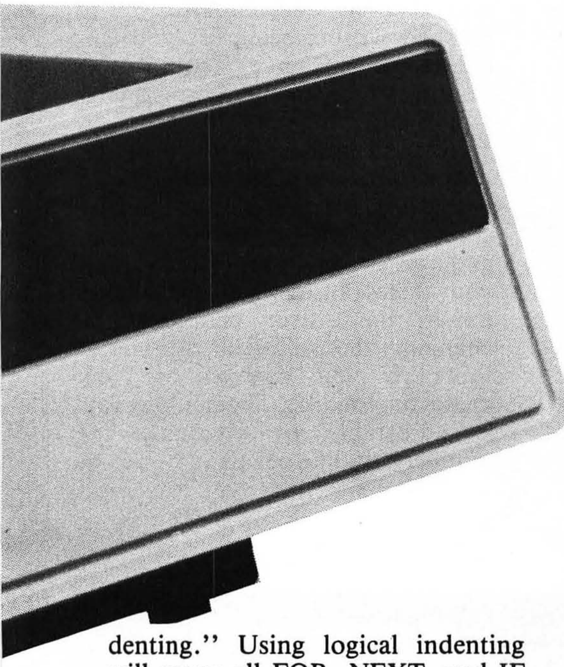
```
IF A <  > - B THEN  PRINT
```

(as normally listed by Applesoft) would become

```
IF A<>-B THEN PRINT
```

Another improvement has been in preventing text from breaking up at the end of a listed line. As with Applesoft, keyword names will be kept whole; in addition, breaks in numbers, strings, and variable names will also be avoided.

There is also an optional formatting method called "logical in-



denting." Using logical indenting will cause all FOR, NEXT, and IF statements to be placed by themselves on separate lines of the listing. The contents of FOR-NEXT loops, and the clauses of IF statements, will be indented to show the change in the logical flow of the program.

To use the list formatter, load the program you want to have listed, insert the disk with the list formatter, turn on the printer, and type "EXEC LFORM". This will cause the BASIC part of the list formatter to be loaded "on top" of the program to be listed, without erasing it. You will be prompted by a number of questions about how the list is to be printed. Each input has a default that you may choose by simply pressing RETURN. Using these *defaults will* generate a proper listing on most systems, so generally all you have to do after typing "EXEC LFORM" is to tweedle the RETURN key.

The first prompt is for the printer slot. Just enter the number of the slot and press RETURN. The default is slot 1. This part of the formatter can be easily altered to accomodate a binary driver if necessary, as it is written in BASIC. (Making such a change is described below.)

The next question is for page length. A length of 66, for example, will result in one line for the header and page number, a blank line, 58 lines of print, and six blank lines for skipping over perforations. If you specify a length of 0, you will get a header with no page number, a blank line, and an uninterrupted listing. Lengths from 0 to 264 may be used, although lengths less than 10 will be considered as 0. The default length is 66, the number of lines on an ordinary 11-inch-long page.

After entering the page length, you must enter the width. It may be best to use a width that is one less than the number of print columns on the printer. This works best with printers that supply an automatic linefeed and return when the end of the line is reached. A range of widths from 25 to 240 is allowed, the default being 79.

Next, the starting and ending line numbers of the list must be specified. If you specify a starting line that is after the beginning of the program, there may be a slight pause before the list is printed, so that the proper logical indentation at that point in the program can be computed. For these two inputs, the defaults are the first and last lines of the program.

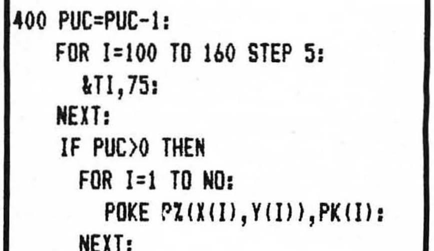
Now the selection for logical indenting must be made. The indenting can be turned off by typing "NO", or activated by entering "YES" or simply pressing RETURN.

The last thing that must be entered before the list is printed is the header. This would most often be the name of the program, and possibly the date. Any characters may be entered, including quotation marks, colons, and commas. Just hit RETURN for a blank header.

If it is necessary to re-enter any of the information requested, a CTRL-C typed for any of the inputs will allow everything to be re-entered. After the list starts to print, a CTRL-C will cause the list to stop, and all output will go back to the screen.

If it is necessary to tie in a binary printer driver, or send special commands to the printer, load the program LFORM.B and modify it as required. Keep in mind that after line 270 nothing should be done that will disturb the input buffer, such as the use of DOS commands or the INPUT statement, because this is where the header is kept.

Sample Listing Generated From *Micro-Man*



```
400 PUC=PUC-1:
    FOR I=100 TO 160 STEP 5:
      &TI,75:
    NEXT:
     IF PUC>0 THEN
       FOR I=1 TO NO:
         POKE P%(X(I),Y(I)),PK(I):
       NEXT:
       POKE P%(X,Y),PK: GOTO 4
405 POKE -16368,0
410 VTAB 24:
    FOR I=1 TO 23:
      PRINT:
      FOR I1=1 TO 200:
      NEXT:
    NEXT:
    VTAB 12: HTAB 12: FLASH: PRINT
    "G A M E  O V E R": NORMAL: VTAB
    22:
    FOR I=1 TO 1000:
    NEXT
415 INPUT "ANOTHER GAME? ";IN$:
     IF IN$="N" THEN
       TEXT: HOME: END
425 GOSUB 2010: GOTO 2
430 FOR I=1 TO NO:
      IF X(I)=X AND Y(I)=Y THEN
        PK=PK(I)
```

K-Byters

REMARK WRITER/ FORMATTER

An Apple K-Byter by Jon Voskuil, Milford, NH

This is a practical application of some of the techniques described in the *Apple F.O.O.L.S. Take-Apart* article elsewhere in this issue. Its function is to write into Applesoft programs neatly-formatted REMark statements which don't LIST like normal REMark statements.

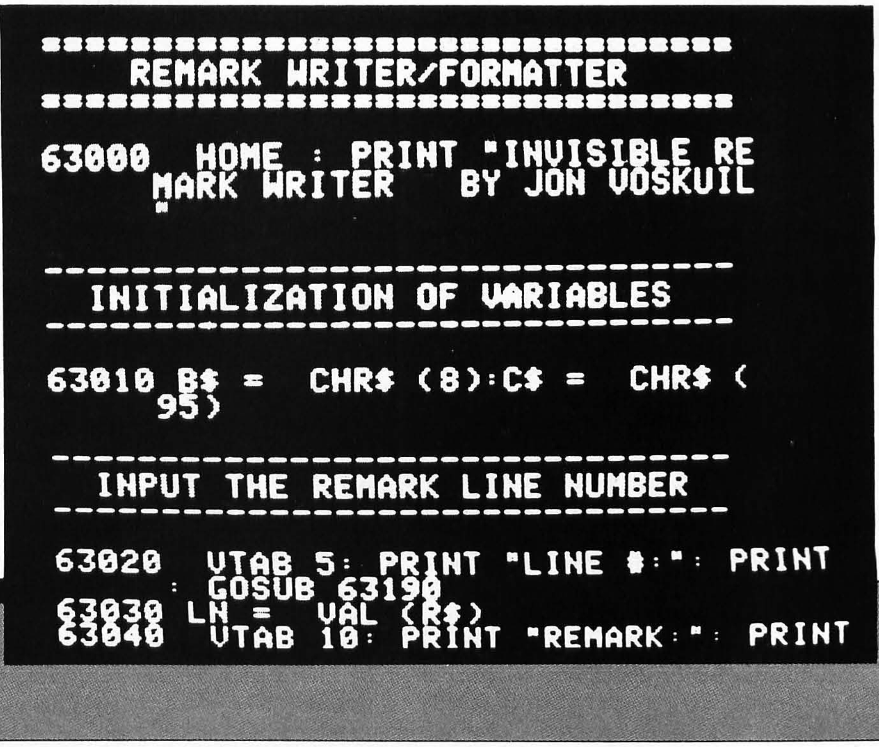
The REM lines written with this program will have two characteristics, and an optional third feature. First, the line number and the word REM will be invisible when the program is LISTed on the screen (unless you use the SPEED= command to slow down the printing rate). Second, the contents of the REM line will be centered in the listing and, if longer than one line, properly broken between words. Third, you ,may optionally choose any character to form a border line above and below the text of the remark.

To make use of the program, it must be temporarily appended to the end of the program you are documenting. If you have a disk, the easiest way to do this is to save the *RemWriter* program as a text file; it can then be EXEC'd at any time into memory along with any other program (as long as it has no line numbers greater than 62999). Lacking a disk, there are two good approaches to using *RemWriter*. Either LOAD it into memɔry before you begin typing in the program you want to document, or use a few POKEs to append it from tape. See the box on the next page for further information on disk and tape appending procedures.

Once appended, the *RemWriter* program can be accessed by typing RUN 60000, and following the prompts. It first asks for the line number of the REMark line to be written, then the text of the REMark, and then the border character (if any). Following this, the screen will clear and you will be instructed to copy over some lines on the screen with the right-arrow key. (Avoid copying over blank spaces beyond the end of the last line.) There is no internal check for maximum allowable length for the REM line; if there are more than six screen lines to be copied, you may need to shorten the text or omit the border. After copying the lines and hitting RETURN, the formatted REMark will have been inserted in your program.

To write another REMark line, simply repeat this procedure, beginning with RUN 60000. After you're finished writing remarks, you can delete *RemWriter* from your beautifully documented program by typing DEL 60000,63270.

You will notice that the program uses an artificially created cursor, a rapidly blinking underline character. This is purely for fun, and can be easily modified to suit your tastes. In line 63010, C$ is set up as the cursor character; by changing the value in the CHR$ function, you can choose any character you like. The blinking rate is adjustable by changing the number 5 in line 63210.



```
*************************************
    REMARK WRITER/FORMATTER
*************************************

63000  HOME : PRINT "INVISIBLE RE
     MARK WRITER   BY JON VOSKUIL
     "


-------------------------------------
  INITIALIZATION OF VARIABLES
-------------------------------------

63010 B$ =  CHR$ (8):C$ =  CHR$ (
     95)

-------------------------------------
  INPUT THE REMARK LINE NUMBER
-------------------------------------

63020  VTAB 5: PRINT "LINE #:": PRINT
     : GOSUB 63190
63030 LN =  VAL (R$)
63040  VTAB 10: PRINT "REMARK:": PRINT
```

APPENDING APPLESOFT PROGRAMS

Here are two procedures for appending one Applesoft program to another, using tape or disk or a combination of both.

The first requires a disk, and makes use of the Apple's EXEC command, which allows you to enter the contents of a text file into memory just as though you had typed the information at the keyboard. To create the text file initially, a one-liner such as the following must be temporarily added to the beginning (or end) of the program you want to be able to append. The value of F$ and the line numbers given in the LIST statement are specifically for *Rem Writer:*

```
1 D$ = CHR$(4): F$ = "REM-
WRITER": PRINT D$ "OPEN"
F$: PRINT D$ "DELETE" F$:
PRINT D$ "OPEN" F$: PRINT
D$ "WRITE" F$: LIST
63000,63270: PRINT D$
"CLOSE": END
```

When you RUN this line, it will write the specified range of lines to disk; and once this is done, the command EXEC REMWRITER (or other program name) will append these lines to whatever program is in memory.

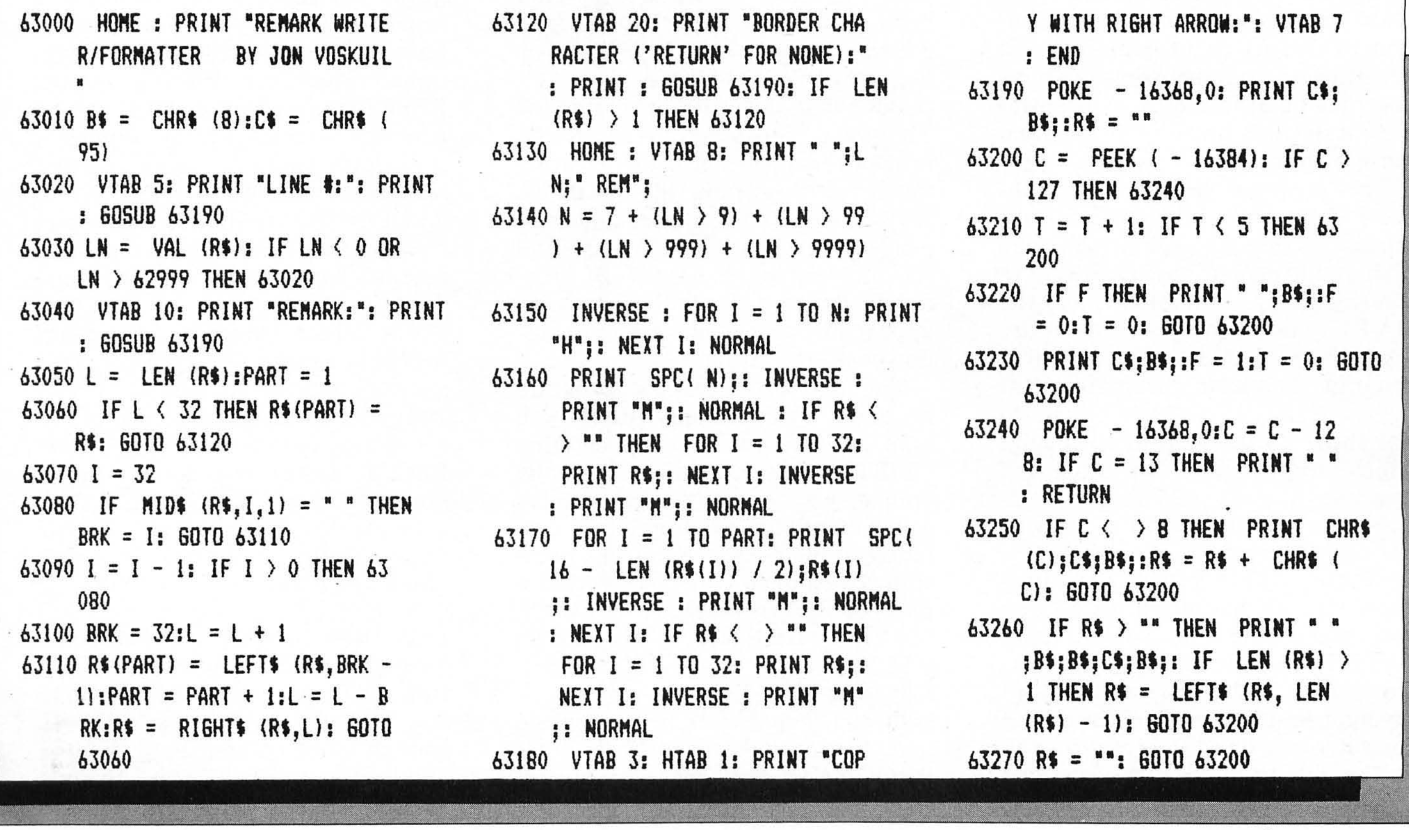
The second procedure will work with tape or disk, or a combination of the two. Here is the necessary series of commands, to be entered from the keyboard:

(a) LOAD (your program)
(b) POKE 103,PEEK(175)-2: POKE 104,PEEK(176)
(c) LOAD *Rem Writer* or other program to be appended)
(d) POKE 103,1: POKE 104,8: DEL 0,0

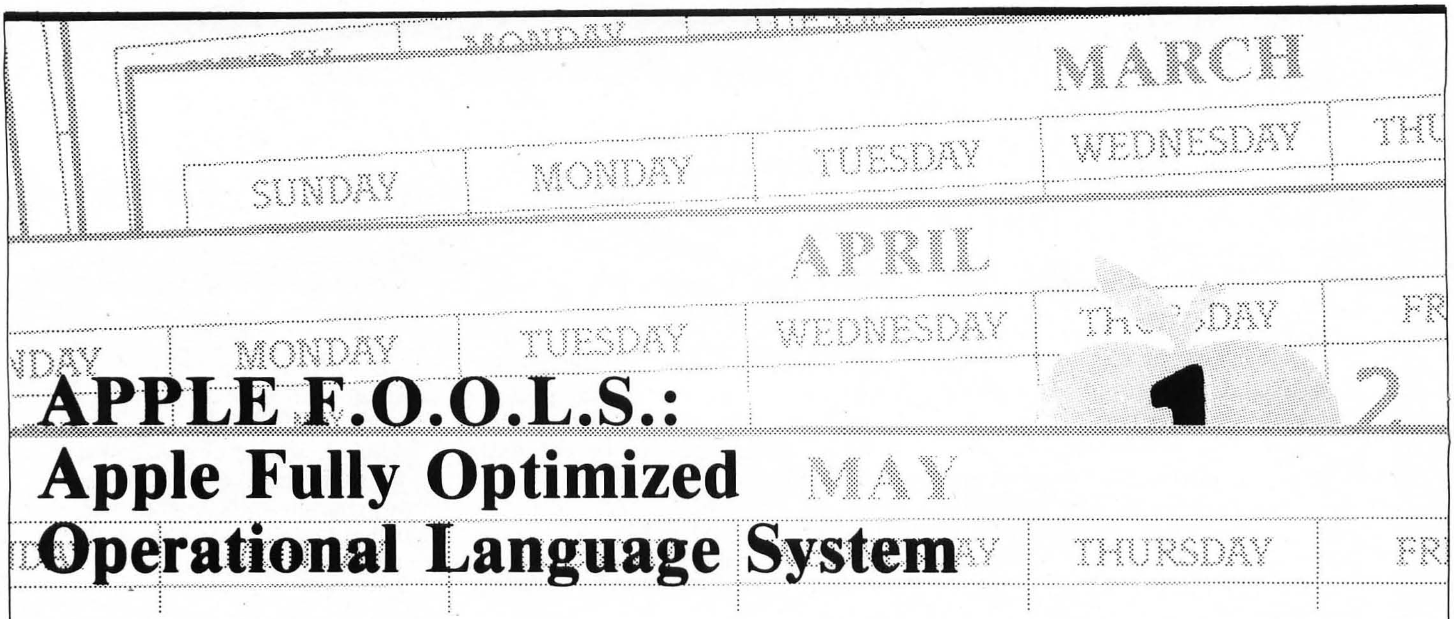
The first pair of POKEs tells the Apple that the beginning of program memory is to be at the end of the program which is already in memory, so that the second program gets LOADed just above the first. Then the second pair of POKEs and the DELete statement restore all the program pointers to their proper positions.

The only time this won't work is when PEEK(175) happens to be a 0 or 1 (which is about 0.78125% of the time), in which case you will get an ILLEGAL QUANTITY ERROR when attempting the first pair of POKEs. If this happens, just add a temporary REM line anywhere in your program and try again. Notice, too, that if you have a line 0 in your program (not generally a good idea in any case), it will be deleted.

This procedure will merge any two BASIC programs from tape or disk, as long as all the line numbers of the program to be appended are higher than those already in memory. (It will work even if this condition isn't met, but will not put the lines in the correct order — which can make for some interesting listings.) To continue appending additional programs to the one in memory, just repeat steps (b) through (d) for each one. ⓢ



```
63000  HOME : PRINT "REMARK WRITE
     R/FORMATTER   BY JON VOSKUIL
     "
63010 B$ =  CHR$ (8):C$ =  CHR$ (
     95)
63020  VTAB 5: PRINT "LINE #:": PRINT
     : GOSUB 63190
63030 LN =  VAL (R$): IF LN < 0 OR
     LN > 62999 THEN 63020
63040  VTAB 10: PRINT "REMARK:": PRINT
     : GOSUB 63190
63050 L =  LEN (R$):PART = 1
63060  IF L < 32 THEN R$(PART) =
     R$: GOTO 63120
63070 I = 32
63080  IF  MID$ (R$,I,1) = " " THEN
     BRK = I: GOTO 63110
63090 I = I - 1: IF I > 0 THEN 63
     080
63100 BRK = 32:L = L + 1
63110 R$(PART) =  LEFT$ (R$,BRK -
     1):PART = PART + 1:L = L - B
     RK:R$ =  RIGHT$ (R$,L): GOTO
     63060
63120  VTAB 20: PRINT "BORDER CHA
     RACTER ('RETURN' FOR NONE):"
     : PRINT : GOSUB 63190: IF  LEN
     (R$) > 1 THEN 63120
63130  HOME : VTAB 8: PRINT " ";L
     N;" REM";
63140 N = 7 + (LN > 9) + (LN > 99
     ) + (LN > 999) + (LN > 9999)

63150  INVERSE : FOR I = 1 TO N: PRINT
     "H";: NEXT I: NORMAL
63160  PRINT  SPC( N);: INVERSE :
     PRINT "M";: NORMAL : IF R$ <
     > "" THEN  FOR I = 1 TO 32:
     PRINT R$;: NEXT I: INVERSE
     : PRINT "M";: NORMAL
63170  FOR I = 1 TO PART: PRINT  SPC(
     16 -  LEN (R$(I)) / 2);R$(I)
     ;: INVERSE : PRINT "M";: NORMAL
     : NEXT I: IF R$ <  > "" THEN
     FOR I = 1 TO 32: PRINT R$;:
     NEXT I: INVERSE : PRINT "M"
     ;: NORMAL
63180  VTAB 3: HTAB 1: PRINT "COP
     Y WITH RIGHT ARROW:": VTAB 7
     : END
63190  POKE  - 16368,0: PRINT C$;
     B$;:R$ = ""
63200 C =  PEEK ( - 16384): IF C >
     127 THEN 63240
63210 T = T + 1: IF T < 5 THEN 63
     200
63220  IF F THEN  PRINT " ";B$;:F
      = 0:T = 0: GOTO 63200
63230  PRINT C$;B$;:F = 1:T = 0: GOTO
     63200
63240  POKE  - 16368,0:C = C - 12
     8: IF C = 13 THEN  PRINT " "
     : RETURN
63250  IF C <  > 8 THEN  PRINT  CHR$
     (C);C$;B$;:R$ = R$ +  CHR$ (
     C): GOTO 63200
63260  IF R$ > "" THEN  PRINT " "
     ;B$;B$;C$;B$;: IF  LEN (R$) >
     1 THEN R$ =  LEFT$ (R$, LEN
     (R$) - 1): GOTO 63200
63270 R$ = "": GOTO 63200
```



APPLE F.O.O.L.S.: Apple Fully Optimized Operational Language System

by Jon R. Voskuil

Apple F.O.O.L.S. **is a utility (?) program for any Apple with Applesoft.**

On last month's Apple Disk and Cassette Versions, subscribers received a program called *Apple F.O.O.L.S.* This "Fully Optimized Operational Language System," as even some who saw only its write-up in the magazine may have guessed, turned out to be an April Fool program. Hardly a profound contribution to the art of programming, it nevertheless contains some features that may be of interest to those who have a devious and slightly warped sense of humor.

The program listing which appears at the end of this article looks innocuous enough. Admittedly, the high and irregular line numbers give it a slightly abnormal look, and the DATA statements would more typically be found at the end of a program than at the beginning. But there are sinister elements which do not show up in the printed listing, which are traps set to catch the unwary.

But I'm getting ahead of myself. The printed listing, just as shown, is the basic *Apple F.O.O.L.S.* program. If you haven't done so yet, you should type it in, SAVE IT TO TAPE OR DISK, and then RUN it. Do not RUN the program without saving it first! If you have entered it correctly and then RUN it, you will not be able to break out of it short of turning off the power to the computer. And you know what that does to all those lines you've just painstakingly typed in.

Assuming that you may need to do some debugging to find typos before the beast will run perfectly, it would be very sensible to add the following line to bypass the RESET-proofing routine:

```
30937 GOTO 30941
```

This will allow the data to be read normally, but will bypass line 30938's crucial "CALL 768" which activates the RESET trap.

But what does the program do? That's the foolish part. When RUN, it has the outward effect of clearing the screen, printing the "APPLE II" logo on the top line, and then printing the Applesoft prompt and flashing cursor a couple of lines down. In short, it makes the screen appear just as it does when you first power up the Apple.

The real effect of the program becomes obvious only when you type in a command. Try entering "LIST", for example. Normally your Apple would obligingly display a list of all the program lines currently in memory. But on this occasion it responds with a beep and a part of its grocery list: "EGGS; MILK; BREAD; BUTTER; PICKLES; ANCHOVIES; SUGAR; FLOUR; . . ."

Or try entering a new program line, such as "10 REM SUPER PROGRAM #99". The response? Another beep, with the accompanying message, "NO PROGRAMS TODAY, I HAVE A HEADACHE." And so it goes: After trying ESC, CTRL-C, and finally RESET (and thinking you've succeeded in escaping at last), you realize that the year 2001 has arrived prematurely and you're no longer in control.

The means employed to accomplished this foolishness are quite straightforward, and make it simple to modify the computer's responses to suit your own tastes in humor. The program is in five sections. The first POKEs in the RESET trapping routine; the second contains several general responses; the third contains more specific responses, along with the commands which are supposed to evoke them; the fourth is the input routine; and the fifth is the main control routine.

The short Machine Language routine which is POKEd into memory and then activated by lines 30936-30940 has the effect of intercepting any attempt at a system RESET. Instead of doing what is normally done when RESET is pressed, this routine causes control to pass to the currently active error-handling routine.

In the program listing, you will find two ONERR GOTO statements. The first of them is a red herring for the unwary, somewhat hidden in line 30950, but obvious enough to be spotted by a snooper and lead him or her astray. The line

63999 which it references is not shown in the program listing; more on that in just a bit.

The real error handling routine is line 54082, which is referenced by the second ONERR GOTO statement, in line 42536. It simply issues a TEXT command and returns to the main program. Visually the effect is the same as a normal RESET, but nothing has really changed.

Lines 30942-30950 DIMension and READ into array N$ a number of general responses which the computer uses when it doesn't have a more specific one in mind. The first DATA item is the number of such responses, and then the responses themselves are listed. Obviously it's an easy matter to add or change these. The computer selects a response randomly from among them when needed.

The next section, in lines 42492-54044, contains most of the substance of the program: specific responses to specific keywords. The first three letters of each of the keywords are READ into the W$ array, and the corresponding responses are READ into the R$ array. The format of the DATA is similar to the previous section, and again is very easy to modify.

Lines 54046-54058 constitute the input routine. It simulates normal Apple operation by GETting each character and checking for various possibilities. The special characters it intercepts are RETURN, ESC, CTRL-X, the forward- and back-arrows, and any other control character. Special messages are printed in response to an ESC or CTRL character anywhere in an input line; other characters produce the same result as they normally would in the immediate mode, and are assembled into the string W$ for later analysis.

The final few lines, 54060-63998, constitute the main program loop. These first print the "APPLE II" logo and then repeatedly call the input routine and analyze the resulting string. If this string is longer than three characters, it is truncated; and then it is compared to the truncated keywords stored in the W$ array. If a match is found, the appropriate message is printed. If a match is not found, a random general message is printed — unless the input string begins with a number, in which case it is interpreted as a program line and suitably squelched.

More Than Meets the Eye

This listing, however, does not show the complete program as it was saved onto the media versions last month. And therein lies more devious subterfuge. Anyone trying to LIST that program received a somewhat less than conventional response from the computer. And I don't mean "EGGS; MILK; BREAD; . . . etc." That's the response you get if you type LIST after RUNning the program; but BEFORE running, it ought to LIST like any normal code. Why didn't it?

The Apple allows you to have quite a bit of fun with program listings. As anyone knows who has imbedded an invisible CTRL-G in a PRINT statement, that statement will beep the speaker every time it LISTs. Beeps can be imbedded in REMarks in exactly the same way — as many of them as you can bear to hear, up to around 240 per line (over 20 seconds of infernal noise). Thus, when attempting to list the media version of *F.O.O.L.S.*, one encounters a great deal of noise interspersed with messages such as "PROGRAM NOT LISTABLE" and "PRESS RESET AND THEN RUN." This is caused by a number of REM lines scattered throughout.

When LISTing the program, the actual line numbers of these interfering lines, and the keyword REM, never show up. The trick behind this is that bells are not the only control characters that can be imbedded invisibly into program lines. Consider what would happen if you could stick a number of CTRL-Hs in a line immediately following the line number and the word REM. CTRL-H is exactly the same as the back-arrow; and as a CTRL-H is LISTed, it backspaces the cursor. Put enough of them in a row, and the cursor will move all the way back to the beginning of the line being LISTed. Then whatever follows in that same REMark line will print right over the line number and word REM, so quickly that it's unnoticeable.

But that's not all. It's just as easy to insert CTRL-Ms (carriage returns) and CTRL-Js (line feeds) into REMs as well. The former will cause the listing of the line to be continued at the first column of the following screen line; the latter also causes a drop down to the next screen line, but without returning to the left edge of the screen.

A suitably doctored REM line, then, might look like the one below. The lower-case b's indicate blank spaces, and the other lower-case letters indicate control characters: g for a CTRL-G, m for a CTRL-M, and h for a CTRL-H.

```
10000  REM hhhhhhhhhhbb
bbbbbbbbmgggPgRgOgGg
RgAgMgmgNgOgTgmgLgIgSg
TgAgBgLgEg!g!g!ggg
```

When this line is LISTed, here's what will happen. First, the characters "10000 REM" will be displayed. Then, the ten CTRL-Hs will cause the cursor to be backspaced to the beginning of the line. Then, ten blank spaces will be printed, wiping out "10000 REM". Next, the CTRL-M will cause the printing position to move down to the first column of the next line; there it will stay while the speaker beeps in response to the three CTRL-Gs. The word "PROGRAM" will then be printed, with more beeps punctuating each letter. The next CTRL-M will again cause a carriage return, and the word "NOT" will be printed on the next line, again with more beeps. Finally, the word "LISTABLE" will be displayed on the next line in the same grating fashion, culminating in yet more beeping.

Obviously, such foolery can be repeated in as many REM lines as you care to intersperse throughout the program — with all sorts of embellishments to further madden the would-be LISTer. Adding a series of about 24 CTRL-Ms or CTRL-Js right after the ten blank spaces, for example, will scroll whatever program lines happen to be displayed on the screen, right off into oblivion. The frustration of trying to read such a listing is compounded by not being able to read what the REM's line number is, in order to delete it. (Unless, of course, you slow down the computer's output by setting SPEED= 10 or some such low number.)

But how does one put control characters into these lines? CTRL-Gs are easy: You just type them in. CTRL-Js can also be typed directly from the keyboard; you can see the effect as you type them. But try typing a CTRL-H or a CTRL-M — what happens? The computer acts just as though you pressed the back-arrow or the RETURN key. The control characters aren't included in the REM statement at all. A different approach is needed.

This approach depends on two things. The first is the Apple's screen editing abilities — the fact that the right-arrow can be used to copy characters directly from screen memory as input into the computer. And the second is the slightly offbeat way in which characters are stored in the Apple's screen memory. Most Apple users are pretty familiar with the former, but may not have much acquaintance with the latter.

When you PRINT a character on the Apple's screen, what happens is that a number representing that character is stored in one of the memory addresses reserved for the video display. The video output circuits interpret that number as a certain character to be displayed on the monitor. As you might expect, the number that's stored in screen memory is related to the ASCII value of the character in question. (See pages 138-139 of the *Applesoft BASIC Programming Reference Manual* for a chart of ASCII values.)

Suppose you PRINT the letter A as the first character on the first line of the screen. This corresponds to memory address 1024. If you then PRINT PEEK(1024), you'll find out what number the computer has stored there to represent that character — 193. Another way of approaching the same thing is to POKE a number into screen memory, and see what character shows up on the screen. Try POKE 1025,194, for example: Lo and behold, the letter B appears at the second position on the top line. The numbers 193 and 194 are the ASCII values of A and B, plus 128.

Well, what happens if you POKE a normal ASCII value (without 128 added) into screen memory? Try it with a command such as POKE 1026,67 (the normal ASCII value for the letter C). The character that appears in the third position on line one is a C, all right — but it's a flashing character rather than a normal one. Conversely, if you PRINT a flashing D at the fourth position on the same line (FLASH: VTAB 1: HTAB 4: PRINT "D";), you'll find that PEEK(1027) yields a 68, the normal ASCII value for D.

Now, the plan is to use the Apple's screen editing features to copy control characters from the screen into program memory. A CTRL-M, for example, has an ASCII value of 13. So POKEing a 13 into screen memory, and then copying over that memory location with the right-arrow as part of a REM line, should do the trick. That sounds like it could be a little complicated, until you try POKEing that 13 into the next screen position (1028) and observe the result: an inverse M.

POKEing an 8 (the ASCII value of CTRL-H) into 1029 gives further reassurance — an inverse H appears next to the inverse M. The conclusion: Copying over an inverse character on the screen will have the effect of copying the corresponding control character into program memory.

Entering REM lines with imbedded control characters, then, is quite a straightforward matter. All you have to do is PRINT the required letters of the alphabet as inverse letters on the screen, and copy over them as needed.

One way of doing this is to construct a PRINT statement which will display the entire program line you need (or at least most of it). Consider line 10000, given above, with the required control characters represented as lower-case g's, h's, and m's. The following command would PRINT on the screen most of the necessary characters to enter this line into memory:

```
PRINT "10000 REM";: INVERSE:
PRINT "HHHHHHHHHH";:
NORMAL: PRINT SPC(10);: IN-
VERSE: PRINT "M";: NORMAL:
PRINT "PROGRAM";: IN-
VERSE: PRINT "M";: NORMAL:
PRINT "NOT";: INVERSE:
PRINT "M";: NORMAL:
PRINT"LISTABLE"
```

After the computer displays this string of normal and inverse characters on the screen, you just move the cursor up to the beginning of the line number and copy it with the right-arrow and REPT keys. The CTRL-Gs are most easily inserted directly from the keyboard as you go along.

Another approach, which avoids the construction of such a complex PRINT statement (but which still requires close attention to what you're doing) is to begin by PRINTing a line full of inverse Ms, and another of inverse Hs, on the screen. Then the cursor can be moved up to these lines to copy one or more characters from them as needed, while entering the REM line from the keyboard.

This is all the knowledge needed to enter most of the offensive REM lines which were included with the media program last month. These are listed together in Figure 1, using the same notation convention as above: Lower-case letters represent control characters, except for lower-case b's which represent blank spaces. Notice that it's important not to type a blank space following the keyword REM when entering the lines, although the Apple will insert a blank space here when LISTING the line.

Figure 1

```
19381 REMhhhhhhhhhhh63999bbPRINTb":HIMEM
:bbHIMEM:bbb":bREMbbbbbbb=bbCOLOR=bb)(bj
bPRINTbbbCOLOR=mSPEED=bbbHGR2bbbbHGR2bbb
PRINTjCOLOR=bjINTbbb))))bb(b)(bbbbFPbbbC
OLOR=bmbbHGRbbbSPEED=jCOLOR=bbbSPC(bbbTA
B(
25158 REMhhhhhhhhhhhbbbbbbbbbbjjjjgggggg
gggggmgPgRgOgGgRgAgMgbgNgOgTgbgLgIgSgTgA
gBgLgE!g!g!g!g!gmmgggggggggggggggmggggg
gggggmgggggggggggggggggg
30935 REMhhhhhhhhhhhNOTICE!!!!!mgggggggg
ggmgPgRgEgSgSgmgRgEgSgEgTgmgAgNgDgmgTgHg
EgNgmgRgUgNg!g!gggggmggggggggggmgggggggg
ggmggggggggggmggggggggggmggggggggggmgggg
gggggmgggggggggggmggggggggggmggggggggggg
gggggggggggggggggggggggggg
30941 REMhhhhhhhhhhhbbbbbbbbbbjjjjjjjjjj
jjjjjjjjjjjjjj
30949 (same as line 30941)
```

```
36712 REMhhhhhhhhhhhhbbbbbbbbbbjjjjjjjjj
jjjjjjjggggggggggmgPgRgOgGgRgAgMgbgNgOg
TgbgLgIgSgTgAgBgLgE!g!g!g!g!gmmmgggggggg
ggggggmggggggggggmgggggggggggggggggggg
ggmggggggggggggggggggggggggggggmggggggg
gggggggggg
42489 (same as line 30935)
42490 REM9&$'%#'&%$(&)(')&'$&%mdRUNbAPPL
EbF.O.O.L.S.
42497 (same as line 30941)
42517 REMhhhhhhhhhhhhbbbbbbbbbbjjjjjjjjj
jjjjjjjjjjjjjjggggggggggggggggggggggggg
gggggggggggggggggggggggggg
42529 (same as line 42517)
48266 (same as line 36712)
54043 (same as line 30935)
54053 (same as line 30941)
54061 (same as line 30941)
59820 (same as line 36712)
63999 PRINT ":bHIMEM:bbHIMEM:bbb":bREMbb
bbbbbb=bbCOLOR=bb)(bjbPRINTbbbCOLOR=mSPE
ED=bbbHGR2bbbbHGR2bbbPRINTjCOLOR=bjINTbb
b))))bb(b)(bbbbFPbbbCOLOR=bmbbHGRbbbSPEE
D=jCOLOR=bbbSPC(bbbTABmdFP
```

Also take note that once one of these ridiculous lines is entered into program memory, it cannot be edited in the usual way, by LISTing it on the screen and copying over it with the right-arrow. Unless you have a utility program such as Neil Konzen's *Program Line Editor* (in which case you can insert control characters much more easily than all this anyway), the line will have to be re-entered in the same way as the original in order to be altered. Of course, any of the lines can be deleted in the normal way, simply by typing the line number and pressing RETURN.

The REM lines are of four general types: beeping messages, multiple linefeeds, apparent garbage, and special disk-system zappers. The beeping messages and multiple linefeeds need no further discussion, and can be identified by their abundance of CTRL-Gs and CTRL-Js.

There are two lines which are constructed specifically to look like garbage — to give the impression that what the observer is seeing really isn't a good LISTing at all. These two — lines 19381 and 63999 — look almost identical. In fact, 19381 LISTs as though it were 63999: Its real line number is over-written to appear as 63999, and is the first line displayed in response to a curious user's LIST command.

The real line 63999 — the dummy ONERR GOTO line mentioned above — not only appears to be garbage, but contains a special surprise for users with disk systems. The plan is that a persistent prying eye will notice the ONERR GOTO 63999 early in the program (after enduring many beeps), and LIST the line to find out what's there. Doing so, one will find what appears to be garbage. Also from that point on, there is no evidence of any program at all being in memory. Why? Because imbedded in this line is a CTRL-D followed by the DOS command "FP".

As I said, the Apple allows you to have lots of fun with listings! If the DOS is booted, and if a CTRL-D is LISTed as the first non-space character of any screen line, the DOS will take notice and will try to interpret the following characters as a DOS command. If a valid command is not found, the listing will stop and the DOS "SYNTAX ERROR" message will be printed on the screen. That's sneaky enough, right there, to stop the casual user from pursuing the quest for a program listing. But even more sneaky is to put a legitimate DOS command following the CTRL-D. Like "FP". Or "RUN APPLE F.O.O.L.S.", as used in line 42490 to surprise (and scare the dickens out of) the user by spinning the disk in the middle of a program listing and RUNning the very program he's trying to figure out. Of course, the possibilities are practically endless for those with a truly sick mind. Just think of what you could do with DOS commands like PR#6, DELETE, and (gasp) INIT!

Playing with unconventional techniques such as these is a lot of fun, and can be an invigorating exercise in wreaking vengeance on your friends (?) who share your interest in computers.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$      APPLESOFT BASIC        $
$     'APPLE F.O.O.L.S.'      $
$   AUTHOR: JON R. VOSKUIL    $
$    (C) 1982    SOFTSIDE     $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Poke in Machine Language RESET trap.

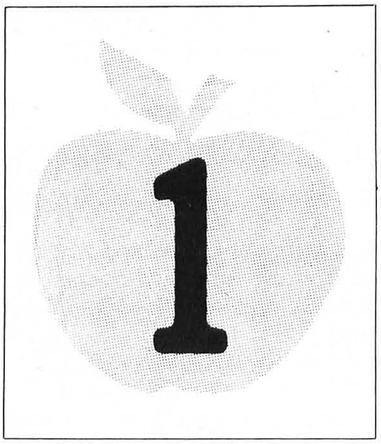
```
30936  FOR I = 768 TO 798: READ J
      : POKE I,J: NEXT
30938  CALL 768
30940  DATA 169,16,141,242,3,169,
      3,141,243,3,73,165,141,244,3
      ,96,104,168,104,166,214,154,
      72,152,72,32,234,3,76,18,212
```

Read general responses into array N$.

```
30942  READ NN
30944  DIM N$(NN)
30946  FOR I = 0 TO NN - 1: READ
      N$(I): NEXT
30948  DATA 6
30950  DATA SAY WHAT??,NO WAY JOS
      E,YOU KNOW THAT'LL NEVER WOR
      K,WOKKA WOKKA,OH YEAH??,WHO
      NEEDS IT?: ONERR  GOTO 63999
```

Read specific keywords and responses into arrays W$ and R$.

```
42492  READ N
42494  DIM W$(N),R$(N)
42496  FOR I = 1 TO N: READ W$(I)
      : READ R$(I): NEXT
42498  DATA 26
42500  DATA CAT,TRY SEARS ROEBUCK

42502  DATA PR#,BOOT IT YOURSELF
42504  DATA RUN,YOU DO THE RUNNIN
      G
42506  DATA LIS,EGGS; MILK; BREAD
      ; BUTTER; PICKLES;      ANCHO
      VIES; SUGAR; FLOUR; ...
42508  DATA LOA,I'M ALREADY LOADE
      D
42510  DATA DEL,I WOULDN'T THINK
      OF IT
42512  DATA NEW,EVERY DAY'S A NEW
      DAY
42514  DATA SAV,WHAT DO YOU WANT
      TO DO THAT FOR?
42516  DATA HOM,HOME IS WHERE THE
      HEART IS
42518  DATA PRI,GET A PENCIL AND
      PAPER
42520  DATA STO,NOT WHEN I'M HAVI
      NG THIS MUCH FUN!!
42522  DATA INI,OVER MY DEAD CPU
42524  DATA BLO,I DON'T WANT THAT
      STUFF IN MY MEMORY
42526  DATA BRU,HOW ABOUT SOMETHI
      NG IN BASIC INSTEAD?
42528  DATA EXE,YOU TOO LAZY TO D
      O IT YOURSELF?
42530  DATA END,NO LET'S KEEP GOI
      NG FOR AWHILE
```



```
42532  DATA HGR,I'M MORE IN A TEX
      T MOOD RIGHT NOW
42534  DATA INV,WHAT'S WRONG WITH
      WHITE ON BLACK??
42536  DATA POK,OH--TRYING THE HE
      AVY STUFF EH?  WELL IT WON'T
      WORK EITHER!!: ONERR  GOTO
      54082
42538  DATA TEX,FOURSCORE AND SEV
      EN YEARS AGO OUR       FATHE
      RS BROUGHT FORTH UPON THIS
              CONTINENT A NEW NATIO
      N CONCEIVED IN     LIBERTY A
      ND DEDICATED TO THE PROPOSIT
      IONTHAT ALL MEN ARE CREATED
      EQUAL....
42540  DATA FLA,WHAT ARE YOU--SOM
      E KIND OF PERVERT?
42542  DATA CAL,I TRIED BUT THERE
      WAS NO ANSWER
42544  DATA FOR,I HATE DOING THE
      SAME THING OVER AND    OVER
      AGAIN
42546  DATA GOT,DON'T TELL ME WHE
      RE TO GO!
42548  DATA HIM,I'LL MANAGE MY OW
      N MEMORY THANK YOU
42550  DATA NOR,I ALREADY AM.  HO
      W ABOUT YOU?
54044 B$ =  CHR$ (7): GOTO 54060
```

Input routine.

```
54046 W$ = "": PRINT "]";
54048  GET C$:C =  ASC (C$): IF C
      = 13 THEN  RETURN
54050  IF C = 27 THEN  PRINT : PRINT
      "THERE IS NO ESCAPE!!!";B$;B
      $;B$: PRINT : GOTO 54046
54052  IF C = 24 THEN  PRINT  CHR$
      (92): GOTO 54046
54054  IF C = 21 THEN  HTAB  PEEK
      (36) + 2: GOTO 54048
54055  IF C = 8 AND  PEEK (36) =
      1 THEN  PRINT : GOTO 54046
54056  IF C = 8 THEN 54058
54057  IF C < 27 THEN  PRINT : PRINT
      "NO!! I'M THE ONE WHO'S IN C
      ONTROL!!!";B$;B$;B$: PRINT :
       GOTO 54046
54058  PRINT C$;:W$ = W$ + C$: GOTO
      54048
```

Main program loop.

```
54060  TEXT : HOME : PRINT  TAB(
      16)"APPLE ]"; CHR$ (91): PRINT

54062  PRINT : GOSUB 54046: PRINT

54066  IF  LEN (W$) < 3 THEN 5407
      2
54068 W$ =  LEFT$ (W$,3): FOR I =
      1 TO N: IF W$ = W$(I) THEN  PRINT
      R$(I);B$: GOTO 54062
54070  NEXT
54072  IF  VAL (W$) > 0 THEN  PRINT
      "NO PROGRAMS TODAY, I HAVE A
      HEADACHE";B$: GOTO 54062
54074  PRINT N$( RND (1) * NN);B$
      : GOTO 54062
54082  TEXT : GOTO 54062
63998  REM
```

YOU ARE INVITED TO PLAY
$
300
300
300
BONUS 800
ALL TARGETS DOWN
WHEN LIT
SCORES 50 AND UP
50
Ring Them BELLS
SPECIAL
WHEN LIT
FLIPPER
FLIPPER
FLIPPER
FLIPPER
MICRO
MAN

MICRO MAN

by George Delp
Apple Version by William Pu,
Translation Contest Winner

***Micro-Man* is an arcade-style game for an Apple with Applesoft and 16K RAM.**

This is a translation of the program "*TRS-Man*" which was published in the January, 1982, issue of ***SoftSide***. It is played on the text screen, with an inverse arrow representing the *Micro-Man* (you). You control him with the five keys centered on the K key: I moves up, M moves down, J moves left, L moves right, and K stops. You must guide the *Micro-Man* through a maze, avoiding the monsters (asterisks). Although the passages of the maze are three units wide, you can travel only in the center of the passages.

As you move *Micro-Man,* he eats the dots which he passes over. There are also a few flashing dollar signs in the maze; eating one of these allows you to eat the monsters. When your time to eat them is almost up, the monsters will start flashing and you will hear a warning. Then, until you eat another dollar sign, the monsters can eat you. Occasionally a monster will become an immobile obstacle in the middle of a passage, making maneuvering more difficult. Once in a while a flashing plus sign will appear; eating it yields an 800-point bonus. Before you finish a board by eating all the dots, you must also eat your quota of monsters. This quota is displayed at the bottom of the screen after the abbreviation "REQ".

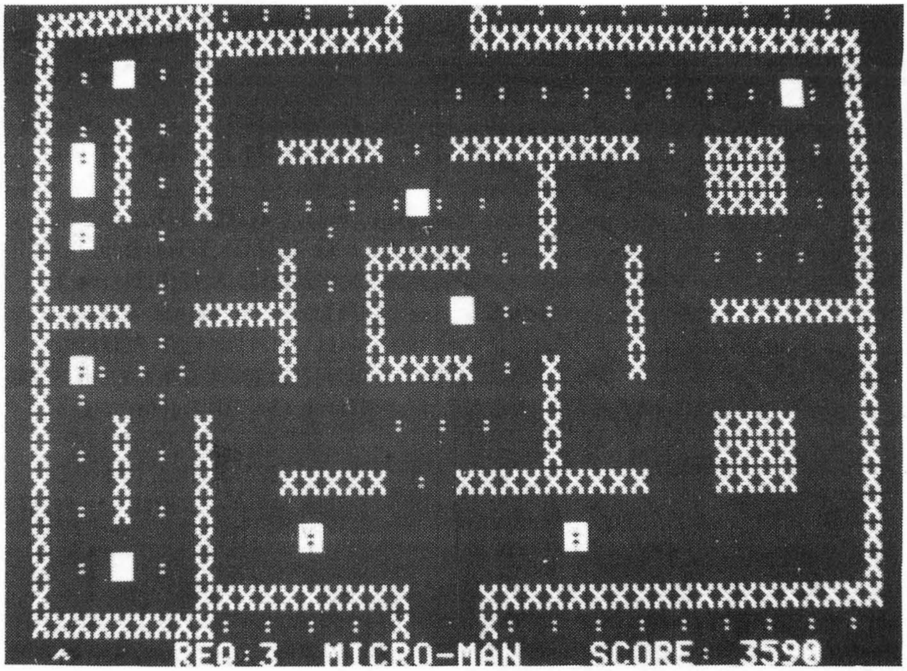
The program senses a collision with the wall by placing an invisible border along all the walls. The character used is CHR$(224). If you prefer a visible border, simply change the variable CO in line 2 to the ASCII value of your choice, as well as all the 224's in lines 2010, 2020, and 2030. (If your Apple has a lower-case adaptor, CHR$(224) may print as a visible character which you may want to change.)

The programming of the game is fairly straightforward. The CALL -868 calls a monitor routine which clears a line of text from the cursor to the righthand edge of the screen. Most of the PEEKs and POKEs are examining and altering the screen memory through the integer array P%. The sound routine was taken from the program "*Space Rescue*," in the February, 1982, *SoftSide*.

Variables

C: Actual number of monsters caught.
CH: Required quota of monsters.
CO: ASCII value of character forming invisible border around walls.
DI%(8): Actual direction of movement of monster.
F1: Time left to eat fruit.
FL%(8): Desired direction to move monster.
HT: Number of dots eaten.
I$,I1$,I2$,I3$: Strings used to print maze.
I,I1,I2: Temporary variables.
IN$: Player input; also used in printing the maze.
INC: Amount to move the monsters.
JU: Your movement increment.
LE: Time during which you can eat the monsters.
NO: Number of monsters.
P%(40,23): Contains value of screen positions.
PK: ASCII value of what *Micro-Man* has eaten.
PK(8): ASCII value of what monsters have covered up.
PUC: Number of *Micro-Men* left.
RT: ASCII value of *Micro-Man.*
S1: ASCII value of flashing asterisks.
SCR: Score.
SH: ASCII value of inverse asterisks.
TI: Time remaining to eat monsters.
X,Y: Your horizontal and vertical position coordinates.
X(8),Y(8): Monsters' horizontal and vertical coordinates.
Z: Your direction of travel.

APPLE™



```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$        APPLESOFT BASIC         $
$          'MICRO-MAN'           $
$     AUTHOR: GEORGE DELP        $
$      TRANSL: WILLIAM PU        $
$     (C) 1982     SOFTSIDE      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Initialization

```
1  GOSUB 4000: GOSUB 2010: DIM X(
     8),Y(I),PK(8),DI%(8),FL%(8)
2 NO = 4:JU = 1:LE = 60:CO = 224:
     S1 = 106:HT = 0:SCR = 0:PUC =
     3:FR = 80:CH = 2
4 X = 19:Y = 20: FOR I = 1 TO 4:X
     (I) = 24 - I:Y(I) = 12:DI%(I
     ) = 1:FL%(I) = I:PK(I) =  PEEK
     (P%(X(I),Y(I))): NEXT :X(5) =
     3:Y(5) = 10:DI%(5) = 1:FL%(5
     ) = 1:PK(5) =  PEEK (P%(X(I)
     ,Y(I)))
6 X(6) = 3:Y(6) = 14:DI%(6) = 1:F
     L%(6) = 1:PK(6) =  PEEK (P%(
     X(6),Y(6))):X(7) = 37:Y(7) =
     10:DI%(7) = 3:FL%(7) = 3:PK(
     7) =  PEEK (P%(X(7),Y(7))):X
     (8) = X(7):Y(8) = 14:DI%(8) =
     3:FL%(8) = 3:PK(8) =  PEEK (
     P%(X(8),Y(8)))
7 RT = 62:SH = 42:INC = 1:PK = 16
     0:Z = 204
9  VTAB 24: HTAB 1: CALL  - 868
12  HTAB 1: VTAB 24: PRINT "
                    ";: FOR I = 1 TO
     PUC: VTAB 24: HTAB I * 2: PRINT
     "^ ";: NEXT : HTAB 8: PRINT
     "REQ:";CH;
15  VTAB 24: HTAB 15: PRINT "MICR
     O-MAN";
18  GOTO 100
```

Movement for monsters.

```
20 I1 =  PEEK (P%(X(I) + INC,Y(I)
     )): IF I1 <  > CO AND I1 <  >
     SH AND I1 <  > S1 THEN  POKE
     P%(X(I),Y(I)),PK(I):X(I) = X
     (I) + INC:PK(I) =  PEEK (P%(
     X(I),Y(I))): POKE P%(X(I),Y(
     I)),SH:DI%(I) = 1: POP : GOTO
     380
23  RETURN
25 I1 =  PEEK (P%(X(I),Y(I) + INC
     )): IF I1 <  > CO AND I1 <  >
     SH AND I1 <  > S1 THEN  POKE
     P%(X(I),Y(I)),PK(I):Y(I) = Y
     (I) + INC:PK(I) =  PEEK (P%(
     X(I),Y(I))): POKE P%(X(I),Y(
     I)),SH:DI%(I) = 2: POP : GOTO
     380
27  RETURN
30 I1 =  PEEK (P%(X(I) - INC,Y(I)
     )): IF I1 <  > CO AND I1 <  >
     SH AND I1 <  > S1 THEN  POKE
     P%(X(I),Y(I)),PK(I):X(I) = X
     (I) - INC:PK(I) =  PEEK (P%(
     X(I),Y(I))): POKE P%(X(I),Y(
     I)),SH:DI%(I) = 3: POP : GOTO
     380
33  RETURN
35 I1 =  PEEK (P%(X(I),Y(I) - INC
     )): IF I1 <  > CO AND I1 <  >
     SH AND I1 <  > S1 THEN  POKE
     P%(X(I),Y(I)),PK(I):Y(I) = Y
     (I) - INC:PK(I) =  PEEK (P%(
     X(I),Y(I))): POKE P%(X(I),Y(
     I)),SH:DI%(I) = 4: POP : GOTO
     380
37  RETURN
```

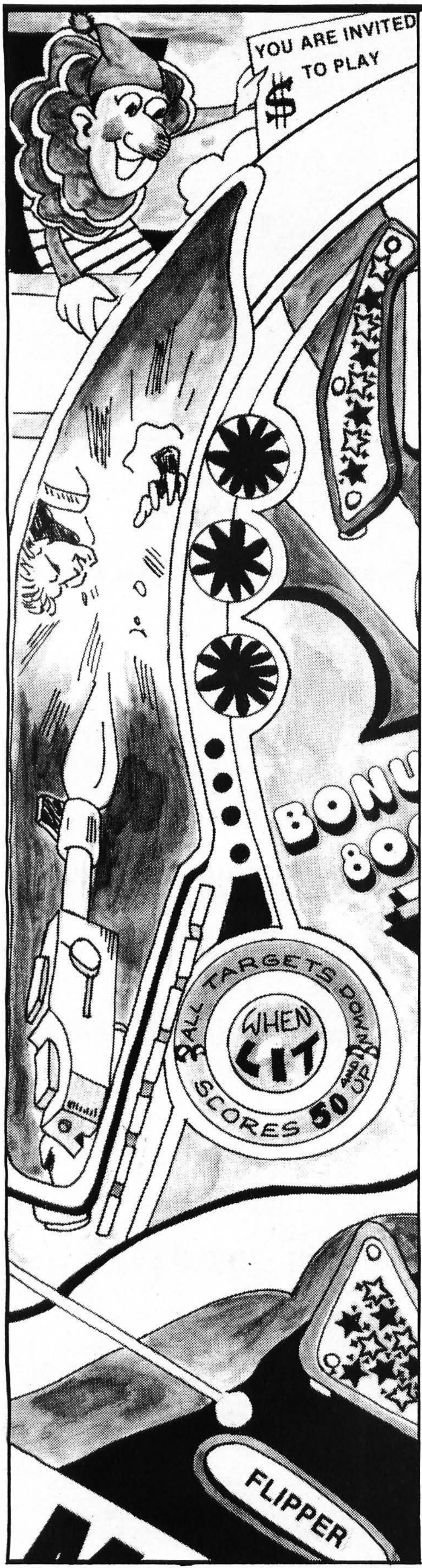
Movement for Micro-Man.

```
50 Y = Y - JU:RT = 30: RETURN
55 X = X - JU:RT = 60: RETURN
60  RETURN
65 X = X + JU:RT = 62: RETURN
70 Y = Y + JU:RT = 22: RETURN
```

Movement logic for monsters.

```
100  FOR I = 1 TO NO
105  IF Y(I) > 22 THEN  POKE P%(X
      (I),Y(I)),PK(I):Y(I) = 2:PK(
      I) =  PEEK (P%(X(I),Y(I))): GOSUB
      390
110  IF Y(I) < 2 THEN  POKE P%(X(
      I),Y(I)),PK(I):Y(I) = 22:PK(
      I) =  PEEK (P%(X(I),Y(I))): GOSUB
      390
205  IF Y = Y(I) THEN FL%(I) = 1 +
      2 * (X < X(I)): GOSUB 390: GOTO
      240
230  IF X = X(I) THEN FL%(I) = 2 +
      2 * (Y < Y(I)): GOSUB 390: GOTO
      240
240  ON FL%(I) GOSUB 20,25,30,35
260  ON DI%(I) GOSUB 20,25,30,35
270 I1 =  SGN ( RND (1) * 4 - 2):
       IF I1 = 0 THEN 270
275 I2 = DI%(I)
280 I2 = I2 + I1: IF I2 = 0 THEN
      I2 = 4
290  IF I2 = 5 THEN I2 = 1
300  IF I2 = DI%(I) OR I2 = FL%(I
      ) THEN 280
310 DI%(I) = I2:FL%(I) = I2
380  NEXT : GOTO 560
```

Collision between monsters and Micro-Man? If yes, check if you can eat.



```
390  IF  ABS (X(I) - X) > 1 OR  ABS
     (Y(I) - Y) > 1 THEN  RETURN

392  IF TI < 1 THEN 395
393  POKE P%(X(I),Y(I)),PK(I):X(I
     ) = 20:Y(I) = 12:DI%(I) = 1:
     FL%(I) = 1:PK(I) =  PEEK (P%
     (X(I),Y(I))): POKE P%(X(I),Y
     (I)),SH:SCR = SCR + ( INT ( RND
     (1) * 6) + 2) * 100:C = C +
     1: FOR I1 = 150 TO 15 STEP  -
     1: & TI1,2: NEXT : IF C = CH
      THEN  VTAB 24: HTAB 12: PRINT
     "F";
394  RETURN
395  POP
```

Micro-Man has died. Either end the game or take away a Micro-Man.

```
400 PUC = PUC - 1: FOR I = 100 TO
     160 STEP 5: & TI,75: NEXT : IF
     PUC > 0 THEN  FOR I = 1 TO N
     O: POKE P%(X(I),Y(I)),PK(I):
      NEXT : POKE P%(X,Y),PK: GOTO
     4
405  POKE  - 16368,0
410  VTAB 24: FOR I = 1 TO 23: PRINT
     : FOR I1 = 1 TO 200: NEXT : NEXT
     : VTAB 12: HTAB 12: FLASH : PRINT
     "G A M E  O V E R": NORMAL :
      VTAB 22: FOR I = 1 TO 1000:
      NEXT
415  INPUT "ANOTHER GAME? ";IN$: IF
     IN$ = "N" THEN  TEXT : HOME
     : END
425  GOSUB 2010: GOTO 2
```

Subroutine called when a collision is detected while the Micro-Man can eat the monsters and the logic section detects the collision.

```
430  FOR I = 1 TO NO: IF X(I) = X
      AND Y(I) = Y THEN PK = PK(I
     )
435  NEXT : RETURN
```

Key input and branch to increment Micro-Man's x and y variables.

```
560 I =  PEEK ( - 16384): IF I >
     200 AND I < 206 THEN Z = I -
     200
580  POKE P%(X,Y),PK: ON Z GOSUB
     50,55,60,65,70:PK =  PEEK (P
     %(X,Y))
```

Check for collision with wall. If collision, move Micro-Man back.

```
590  IF PK = CO OR PK = 32 OR PK =
     SH THEN JU =  - 1: ON Z GOSUB
     50,55,60,65,70:PK =  PEEK (P
     %(X,Y)): POKE P%(X,Y),RT:JU =
     1: GOTO 100
```

Draw Micro-Man.

```
595  POKE P%(X,Y),RT: IF PK = 160
      THEN 620
```

Micro-Man has eaten something. Check and act.

```
600  IF PK = 186 THEN PK = 160: FOR
     I = 230 TO 20 STEP  - 25: &
     TI,2: NEXT :SCR = SCR + 10:H
     T = HT + 1: IF HT > 104 THEN
     800
605  IF PK = 42 OR PK = 106 AND T
     I > 0 THEN  GOSUB 430: GOTO
     620
610  IF PK = 100 THEN PK = 160:IN
     C =  - 1:TI = LE:LE = LE - 4
     : FOR I = 8 TO 4 STEP  - 2: FOR
     I1 = 230 TO 20 STEP  - 15: &
     TI1,I: NEXT : NEXT :SH = 42:
      IF LE < 10 THEN LE = 10
615  IF PK = 107 THEN SCR = SCR +
     800:I1 = 2: FOR I = 230 TO 2
     00 STEP  - 1: & TI,I1:I1 = I
     1 + 1: NEXT I:PK = 160:F1 =
     1
```

Decide bonus and time during which you can eat monsters.

```
620 TI = TI - 1: IF TI > 0 AND TI
     < 10 THEN SH = 106: & T50,2
     0: IF TI = 1 THEN SH = 42:IN
     C = 1
625 F1 = F1 - 1: IF F1 = 0 THEN  POKE
     P%(19,8),160
```

Wrap-around Micro-Man.

```
630  IF Y > 22 THEN  POKE P%(X,Y)
     ,PK:Y = 2:PK =  PEEK (P%(X,Y
     )): GOTO 100
```

```
640  IF Y < 2 THEN  POKE P%(X,Y),
     PK:Y = 22:PK =  PEEK (P%(X,Y
     )): GOTO 100
```

Check if time to initialize bonus.

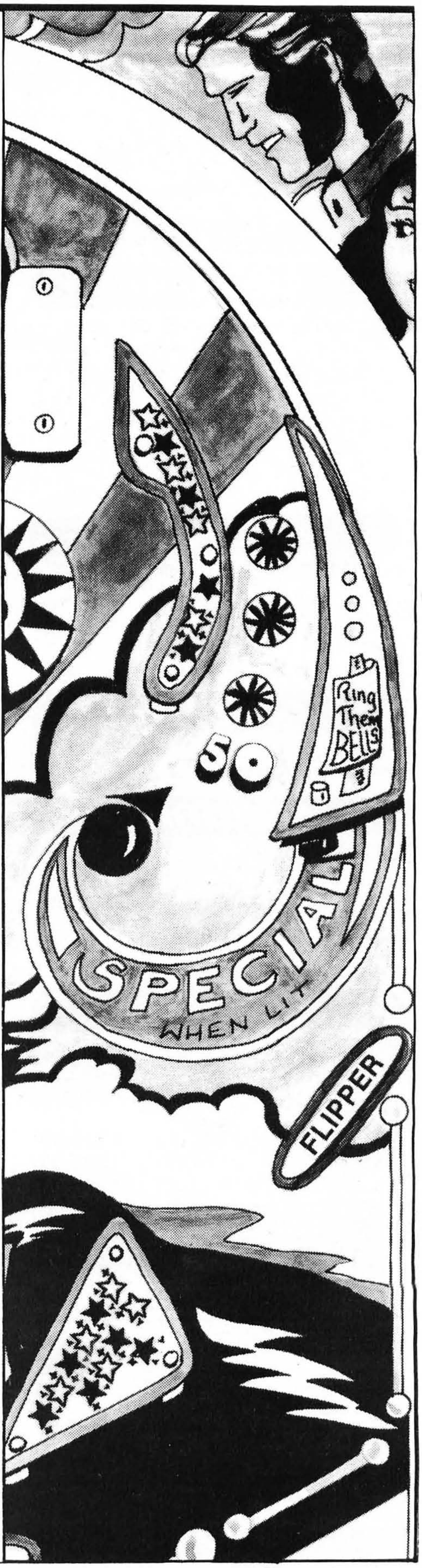
```
650  IF HT = 52 THEN F1 = FR: POKE
     P%(19,8),107: IF FR > 20 THEN
     FR = FR - 10
660  HTAB 27: VTAB 24: CALL  - 86
     8: PRINT "SCORE: ";SCR;
670  GOTO 100
```

Board cleared. Check if met requirement. Draw new board.

```
800  IF CH > C THEN  PRINT : PRINT
     : PRINT : PRINT : PRINT "BEC
     AUSE YOU DID NOT CATCH YOUR
     QUOTA OF     GHOSTS, YOU HAV
     E LOST!!!": FOR I = 1 TO 17:
      PRINT : FOR I1 = 1 TO 300: NEXT
     : NEXT : FLASH : HTAB 15: VTAB
     12: PRINT "G A M E  O V E R"
     : NORMAL : VTAB 22: GOTO 415

810  FOR I = 235 TO 20 STEP  - 1:
      & TI,2: & TI + 20,2: & TI,2
     : & TI - 20,2: PRINT "    ";
     : NEXT
815 CH = CH + 1:HT = 0:NO = NO +
     1:TI =  - 1:C = 0: IF NO > 8
      THEN NO = 8
820  GOSUB 2010
830  GOTO 4
```

Print maze.

```
2010 I$ =  CHR$ (224): FOR I = 1 TO
     5:I$ = I$ + I$: NEXT :I2$ =
      CHR$ (224) + "X" +  CHR$ (2
     24):I1$ = "X": FOR I = 1 TO
     5:I1$ = I1$ + I1$: NEXT
2013 I3$ = ": ": FOR I = 1 TO 4:I
     3$ = I3$ + I3$: NEXT
2014  HOME
2015  FOR I = 1 TO 12: ON I GOSUB
     2020,2030,2040,2050,2060,207
     0,2080,2090,2100,2110,2120,2
     130: HTAB 1: VTAB I: PRINT I
     N$;: VTAB 24 - I: HTAB 1: PRINT
     IN$;: NEXT
2016  FLASH : VTAB 3: HTAB 5: PRINT
     "$": HTAB 37: PRINT "$";: VTAB
     20: HTAB 37: PRINT "$": HTAB
     5: PRINT "$": NORMAL
2017  POKE P%(21,12),32: RETURN
2020 IN$ =  LEFT$ (I1$,9) +  LEFT$
     (I3$,8) + "X" +  CHR$ (224) +
      CHR$ (160) +  CHR$ (224) +
     "X" +  LEFT$ (I3$,18): RETURN

2030 IN$ =  RIGHT$ (I2$,2) +  RIGHT$
     (I$,6) +  RIGHT$ (I1$,10) +
      CHR$ (224) +  CHR$ (160) +
      CHR$ (224) +  RIGHT$ (I1$,1
     9): RETURN
2040 IN$ =  RIGHT$ (I2$,2) +  LEFT$
     (I3$,5) + I2$ +  LEFT$ (I$,9
     ) +  CHR$ (186) +  LEFT$ (I$
     ,18) +  LEFT$ (I2$,2): RETURN

2050 IN$ =  RIGHT$ (I2$,2) +  CHR$
     (160) +  LEFT$ (I$,3) +  CHR$
     (160) + I2$ +  LEFT$ (I3$,26
     ) +  CHR$ (160) +  CHR$ (186
     ) +  LEFT$ (I2$,2): RETURN
2060 IN$ =  RIGHT$ (I2$,2) +  CHR$
     (186) + I2$ +  CHR$ (186) +
     I2$ +  CHR$ (160) +  LEFT$ (
     I$,7) +  CHR$ (160)
2065 IN$ = IN$ +  LEFT$ (I$,11) +
      CHR$ (160) +  LEFT$ (I$,6) +
      CHR$ (160) +  LEFT$ (I2$,2)
     : RETURN
2070 IN$ =  RIGHT$ (I2$,2) +  CHR$
     (160) + I2$ +  CHR$ (160) +
     I2$ +  CHR$ (186) +  LEFT$ (
     I2$,2) +  LEFT$ (I1$,3) +  RIGHT$
     (I2$,2) +  CHR$ (186) +  LEFT$
     (I2$,2)
2075 IN$ = IN$ +  LEFT$ (I1$,7) +
      RIGHT$ (I2$,2) +  CHR$ (186
     ) +  LEFT$ (I2$,2) +  LEFT$
     (I1$,2) +  RIGHT$ (I2$,2) +
      CHR$ (186) +  LEFT$ (I2$,2)
     : RETURN
2080  GOSUB 2060:IN$ =  LEFT$ (IN
     $,24) + "X" +  MID$ (IN$,26,
     7) +  RIGHT$ (I1$,4) +  RIGHT$
     (IN$,4): RETURN
2090 IN$ =  RIGHT$ (I2$,2) +  CHR$
     (160) + I2$ +  CHR$ (160) +
     I2$ +  RIGHT$ (I3$,13) + I2$
      +  LEFT$ (I3$,5):IN$ =  LEFT$
     (IN$,31) +  LEFT$ (I2$,2) +
      LEFT$ (I1$,2) +  RIGHT$ (I2
     $,2) +  CHR$ (186) +  LEFT$
     (I2$,2): RETURN
2100 IN$ =  RIGHT$ (I2$,2) +  CHR$
```



```
     (186) +  RIGHT$ (I$,3) +  CHR$
     (186) +  RIGHT$ (I$,3) +  CHR$
     (186) +  RIGHT$ (I$,3) +  CHR$
     (186) +  RIGHT$ (I$,7) +  CHR$
     (160)
2105 IN$ = IN$ + I2$ +  CHR$ (160
     ) +  RIGHT$ (I$,3) +  CHR$ (
     160) +  RIGHT$ (I$,6) +  CHR$
     (160) +  LEFT$ (I2$,2): RETURN

2110 IN$ =  RIGHT$ (I2$,2) +  RIGHT$
     (I3$,9) + I2$ +  CHR$ (160) +
      LEFT$ (I2$,2) +  RIGHT$ (I1
     $,3) +  RIGHT$ (I2$,2)
2115 IN$ = IN$ +  CHR$ (186) + I2
     $ +  CHR$ (186) + I2$ +  LEFT$
     (I3$,8) +  LEFT$ (I2$,2): RETURN

2120 IN$ =  RIGHT$ (I2$,2) +  RIGHT$
     (I$,4) +  CHR$ (186) +  RIGHT$
     (I$,4) + I2$ +  CHR$ (186) +
     I2$ +  RIGHT$ (I$,4)
2125 IN$ = IN$ +  CHR$ (160) +  RIGHT$
     (I$,3) +  CHR$ (160) + I2$ +
      CHR$ (160) +  RIGHT$ (I$,7)
      +  LEFT$ (I2$,2): RETURN
2130 IN$ =  RIGHT$ (I1$,4) +  RIGHT$
     (I2$,2) +  CHR$ (160) +  LEFT$
     (I2$,2) +  RIGHT$ (I1$,3) +
      RIGHT$ (I2$,2) +  CHR$ (160
     ) + I2$ +  CHR$ (160)
2135 IN$ = IN$ +  CHR$ (160) +  CHR$
     (160) +  CHR$ (160) +  LEFT$
     (I3$,5) + I2$ +  CHR$ (186) +
      LEFT$ (I2$,2) +  RIGHT$ (I1
     $,7): RETURN
```

Initialize P%(40,23).

```
4000  GOSUB 6000: DIM P%(40,23)
4010  FOR I = 1 TO 40: FOR I1 = 1
      TO 8:P%(I,I1) = 1023 + (I1 -
     1) * 128 + I: NEXT : NEXT
4020  FOR I = 1 TO 40: FOR I1 = 9
      TO 16:P%(I,I1) = 1063 + (I1
      - 9) * 128 + I: NEXT : NEXT

4030  FOR I = 1 TO 40: FOR I1 = 1
     7 TO 23:P%(I,I1) = 1103 + (I
     1 - 17) * 128 + I: NEXT : NEXT
```

Initialize sound routine.

```
5000  FOR I = 1 TO 66: READ I1: POKE
     I + 767,I1: NEXT
5040  POKE 1013,76: POKE 1014,0: POKE
     1015,3: FOR AZ = 1 TO 5000: NEXT

5050  RETURN
5100  DATA 201,084,208,015,032,17
     7,000,032,248,230,138,072,03
     2,183,000,201,044,240,003,07
     6,201,222,032,177,000,032,24
     8,230,104,134,003,134,001,13
     3,000
5110  DATA 170,160,001,132,002,17
     3,048,192,136,208,004,198,00
     1,240,007,202,208,246,166,00
     0,208,239,165,003,133,001,19
     8,002,208,241,096
```

Print title page.

```
6000  TEXT : HOME : PRINT : HTAB
     11: INVERSE : PRINT "M I C R
      O  -  M A N": NORMAL : PRINT
     : PRINT  TAB( 15)"BY BILL PU
     ": PRINT : PRINT : INVERSE
6010  HTAB 9: PRINT ">";: NORMAL
     : PRINT " . . .  THE MICRO M
     AN": PRINT  TAB( 16)"CONTROL
     S": PRINT : HTAB 20: INVERSE
     : PRINT "I": NORMAL : HTAB 2
     0: PRINT ".": HTAB 16: INVERSE
     : PRINT "J   K   L";
6015  HTAB 17: NORMAL : PRINT " .
      ";: HTAB 21: PRINT " . ": PRINT
      TAB( 20);".": INVERSE : HTAB
     20: PRINT "M"
6017  PRINT
6020  HTAB 10: PRINT "*";: NORMAL
     : PRINT " . . .  THEM  . . .
      200-800"
6030  INVERSE : HTAB 10: PRINT "+
     ";: NORMAL : PRINT " . . .
     BONUS . . . 800"
6040  HTAB 10: FLASH : PRINT "$";
     : NORMAL : PRINT " . . .ENER
     GIZER . . 50"
6050  HTAB 10: PRINT ": . . .  DO
     TS  . . . 10"
6060  PRINT : PRINT "REQ: THE NUM
     BER OF THEM WHICH YOU MUST":
      PRINT  TAB( 14)"EAT PER BOA
     RD"
6065  PRINT
6070  PRINT  TAB( 7);: FLASH : PRINT
     "INITIALIZATION IN PROGRESS"
     ;: NORMAL : RETURN
```

Electric Duet

A review by Jon Voskuil

By Paul Lutus. From Insoft, 10175 Barbur Blvd., Suite 202B, Portland, OR 97219. Requires an Apple II with at least 32K RAM and DOS 3.3. Retail price: $29.95.

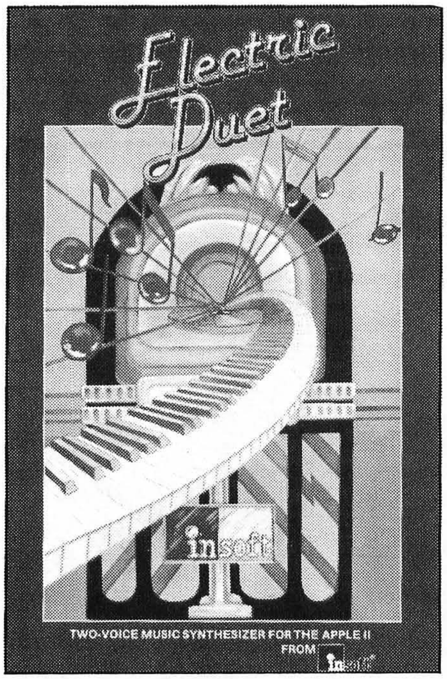
In the background as I write this I hear strains of Bach, Mozart, and Scott Joplin. Between each composition I hear the momentary whir of a disk drive. Glancing over at my monitor, I see highlighted the name of the piece being played inside a screen border wrapped around with the words "Apple II Electronic Jukebox."

It's really not very unusual to hear beeps, blasts, tones, and even tunes emanating from Apples. The Apple programs which appear in **SoftSide** often contain sound and musical enhancements, as do most commercially-available programs (especially of the arcade-game genre). Last November's issue was devoted almost exclusively to the use of music in microcomputers (and vice versa).

What IS quite unusual (unless you've invested in some additional hardware) is to hear two simultaneous voices coming from an Apple. A single musical voice is nice. It lets you play perfectly recognizable fanfares for winners and dirges for losers, and even do a respectable job accompanying flying bumblebees and other such classic solo creatures. But after awhile one really longs for some good ol' fashioned harmony. *Electric Duet* makes it happen — without any hardware additions at all.

Electric Duet is a two-voice music synthesizer/editor which is a well-conceived and neatly-done software package that goes a long way toward making it easy to create tunes in two-part harmony. Upon booting the disk, the initial temptation (to which you should definitely give in) is to enter the "jukebox" mode and listen to the 14 compositions already programmed and saved on the disk. You can settle for hearing these through your Apple's built-in speaker, or opt for much better sound by routing the program's output through your cassette cable and into an external amplifier and speaker. As you listen, you'll be able to hear several of the six different tonal qualities which can be programmed for each of the voices.

By the way, the word "voice" sometimes refers simply to one of the notes to be played, and sometimes refers to the particular sound quality of a played note (whether it sounds like an oboe, a violin, etc.). In the first sense, this is a two-voice music synthesizer, because you can play two different notes simultaneously, in harmony with one another. In the second sense, the program gives you a choice of six voices or sounds which can be assigned to either of the two notes being played.



After just listening for awhile, you'll probably get a yen to do something more creative. Switching over to the "piano" mode, you have the chance to play two-voice music in real time on the Apple keyboard. The screen display simulates a piano keyboard, showing you which keys to press to get the desired notes. The two upper rows of keys play one voice, the two lower rows the other voice. Definitely not a mode for virtuoso performances, this nevertheless allows you to play around with tunes and harmonies to see how they might sound in an actual composition.

Serious composing requires switching over to the music editor, which offers you two different modes of note entry. The first allows keypunched input of values for each note's duration and pitch, in a very natural format. For example, a half-note C in the third octave would be entered with the key sequence, "2,3C". Pressing ESC toggles to the other note to be entered, and then RETURN moves you to the next pair of notes. The notation for the previously entered pair of notes also remains on the input line; this allows you to change only the parts of the notation that need to be changed, without having to re-enter a duration or pitch which remains the same.

The second method of entering notes into the editor is to play them on the keyboard just as you do in the piano mode. Whatever "piano" key you press will play the note and cause the correct notation to be displayed on the input line; and pressing RETURN will enter that note and move on to the next one. You can set the duration of each note as you go, or (more easily) enter a series of notes and then go back to correct the durations. The documentation suggests playing notes in one voice for awhile, then going back to play notes in the accompanying voice for awhile. This procedure works quite nicely.

In addition to the ability to scroll backward and forward through a file of notes (listening to each one if you choose) to make changes in any given line, you can also easily delete or insert lines in any position. The procedure is to scroll to the desired line, "open" the file at that point with a couple of keystrokes, delete or insert the line(s), and then close the file with a couple more keystrokes. In addition to adding forgotten notes, inserting voice changes at various places would be another common application for this ability. And, the same procedure allows you to insert into your file a series of notes previously saved onto disk — a natural for entering musical refrains and other repetitive material.

The editor also enables you to make limited or universal changes to the notes in your note file, using the "transpose" function. You can transpose the pitch of any group of notes, up or down, any chosen number of half-steps. You can do the same with their duration, shortening or lengthening them by any number of increments (e.g., from eighth to dotted eighth to quarter to dotted quarter...). Since this function acts on only one of the two voices at a time, you can independently manipulate the relative pitch of the two voices.

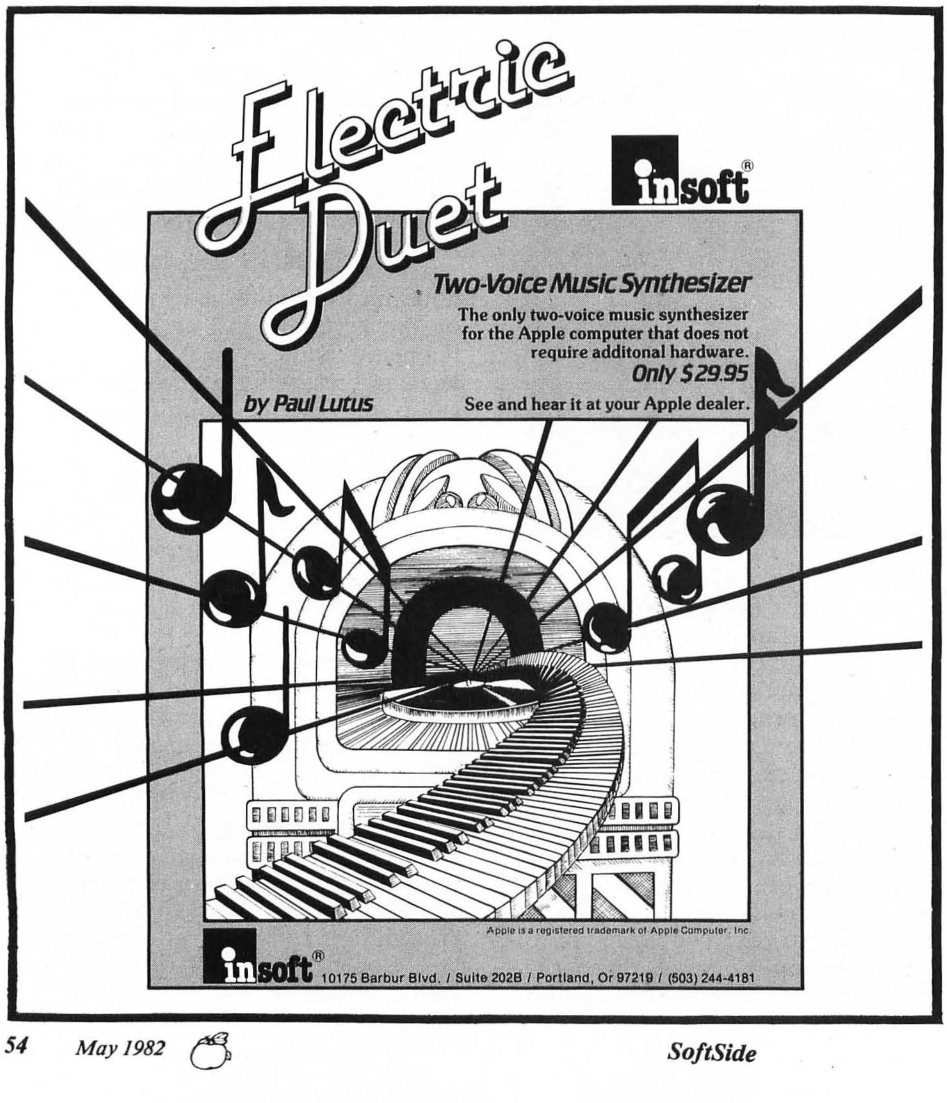
One of the best things about *Electric Duet* is that you can freely transplant the music you create with it into your own programs. (The publisher asks only that you include an acknowledgement of the *Electric Duet* program — a perfectly reasonable request.) The music files themselves are saved in binary form onto any 16-sector disk. The only other thing required is the Machine Language routine to play them, which can also be written from the *Electric Duet* program onto a normal disk. This routine can be adjusted to play music either through the Apple's speaker or through its cassette port, and can be located at any convenient place in memory (such as 300 hex). Once the player program and the music file(s) are on your disk, all you need to do is a couple of BLOADs, two POKEs, and a CALL, and your very own program is playing two-part harmony.

I find it pretty hard to fault *Electric Duet*. Its commands and procedures are logical and simple, easily learned but versatile enough to do the job. Prompt lines and menus combine to let you know what the computer is expecting as input, and errors are well trapped. Accidentally pressing RESET causes no problem; you're simply returned to the main menu with no damage to anything in memory. The documentation (a 17-page booklet plus a page of errata) is not dressed up in a fancy wrapper, but is clear and adequate. The single copy-protected disk supplied is double-sided, providing a backup of sorts if one side becomes defective. (I say "of sorts," because dogs, magnets, and other such perils seldom damage only one side of a disk.)

"One of the best things about Electric Duet is that you can freely transplant the music you create with it into your own programs."

The one thing I do find truly objectionable is that, once booted, there is no way to exit the program short of turning off the power switch. There ought to be a more graceful finale than that.

In spite of that one annoyance, I have no hesitation recommending *Electric Duet*. It's a good program at a reasonable price, and should be very useful — both in its own right as a creative program, and in providing good musical enhancements to other programs.



A Fix For The Apple Microtext Tape Problem

Note from the author of *Microtext:*

As mentioned in the April issue, the Apple version of the Microtext word processor program had a fatal problem with the tape save/load routine, which was badly in need of an elegant solution. In late February, just a few days too late for inclusion in that issue, a friend dug up what appeared to be the needed information to create a successful tape routine. Ollie Holt, president of our local Apple user's group, located an article in Contact '78 describing how string arrays could be written to tape and recalled as needed.

Just as I was getting around to incorporating the techniques described in the article into Microtext 1.2, *I received in mid-March a letter from reader Tom Pollard of Dunstable, Massachusetts, with his own proposed solution. Tom suggested an approach similar to the one Ollie located, with some refinements. He supplied a Machine Language routine and complete line listings for revising* Microtext, *along with a good explanation of the problem and the solution.*

So with thanks to both Ollie and Tom, we're happy to publish the letter and modifications which follow, which now provide Apple tape users with a workable word processor. (All references to Microtext 1.1 *apply to version 1.2 as well.)*

Dear ***SoftSide***;

Representing the 10% of Apple owners that do not have a disk drive, there is a serious problem with the tape save and load function of your *Microtext 1.1*. The STORE and RECALL in Applesoft are only for real and integer arrays. String arrays must be converted to an integer array using the ASC function in order to be stored (*Applesoft Reference Manual*, page 63). Unfortunately an array would have to be dimensioned 1000 by 37. Each variable takes two bytes of memory. So 1000 times 37 times 2 equals 74K bytes of storage. As it stands now, if you save a letter to cassette, then turn off your Apple, and then power up and try to recall your letter off cassette, all you will get is garbage. What follows explains why this occurs, and how to fix it.

What Went Wrong

To understand this we must examine how Applesoft maps its arrays in memory. Applesoft programs are stored from $800 up. Locations $69,$6A point to the end of the BASIC program, and the start of simple variable space (nondimensioned variables). The memory from the end of the Applesoft program to $C000 (in a 48K Apple, without disk) or $9600 (with disk) is used for variables. $6B and $6C point to the end of simple variables, and to the start of array variables. $6D and $6E point to the end of variable storage. If X were dimensioned as X(3,3) Apple would look at $6D,$6E and store as in Figure 1A, and change $6D,$6E to the end of X(3,3) storage. Figure 1B shows X dimensioned as X%(3,5).

Now we come to the string array. Figure 1C shows X$(7,3). For real and integer variables the data blocks are the variable's actual value. But for string arrays, the data blocks are more like a look-up table for string variables; because with string variables, the data blocks are pointing to string storage data. The string storage is from HIMEM ($C000 or $9600) down. The point is that a STORE X$ command stores only this look-up table, not the actual string data.

How to Solve This Problem

Well, I had the same problem with my mailing list of 200 possible names. There would have to be N%(200,40) for the names, A%(200,40) for the addresses, and C%(200,40) for the city, state, and zip. A total of 200 times 40 times 2 times 3, or 48K bytes worth of storage. I could only use one integer variable array to store three string arrays. It went like this: Convert to integer, start tape, save to tape, stop tape; convert to integer, start tape, save to tape, stop tape; etc. Very wasteful; but about the only answer.

continued on next page

Then I saw your problem. What had to be done was to save the "look-up table," the start and finish addresses of the string storage, and the string storage data itself. A machine language subroutine was needed for this solution.

To write to tape: After you have STOREd all your string variables ("look-up table"), you must CALL 768 to save the string storage and HIMEM pointers, and the string data itself. Listing 1 is an example.

To read from tape: After using RECALL to retrieve all string variable "look-up tables," CALL 800 to place the string storage at an unused area in memory below string storage already in use. The problem arose, that a 48K cassette-only Apple's string storage area was from $C000 down, and the same Apple using DOS had string storage from $9600 down. If the cassette Apple transferred its string storage to a disk Apple, it would bomb the DOS. Some tricks had to be done. We now look at $6F,$70 (end of string storage) and load down.

The FIX subroutine: Now the RECALLed string addresses are still pointing to their old location (maybe in DOS), so an offset must be added to each address. They will then point to the string storage at its new location. A fix routine must be used for each string "look-up table" recalled. Listing 2 shows how to recall A$, B$, and C$. The fix routine simply finds the difference between the old string storage pointer and the new string storage pointer; then starts at the bottom of the array (X$(0,0,0)), and changes all active string variable addresses to the new location.

This may all be a lot of work, but this is the only way I can think of solving the problem. Listing 3 is all the changes to be made to *Microtext 1.1* to be a 1.1.1 version. (This letter is on cassette using the new *Microtext 1.1.1*).

One danger with this Machine Language program (if used with other programs) is that strings MUST be made either by inputting characters or by calculation. If the statement A$(0) = "HI" were used instead of the statement A$(0) = "H" + "I", the string look-up table would point to the program instead of the string storage area. When later RECALLed, with the added offset, it would point elsewhere in the program. Listing 1 contains such examples.

References: *Applesoft Reference Manual*, page 140; *Softalk*, February 1982, page 120.

Sincerely,

Tom Pollard

Tom Pollard

Listing 1

```
50  FOR Z = 768 TO 932: READ A: POKE
     Z,A: NEXT
100  DIM A$(100)
110  DIM B$(100,2)
120  DIM C$(100,2,1)
125  REM  STRINGS MUST BE MADE
     EITHER BY INPUTTING CHARAC-
     TERS OR BY CALCULATION. IF
     A$(0)="HI" WERE USED THE
     STRING "LOOK-UP TABLE" WOULD
     POINT TO THE PROGRAM INSTEAD
     OF THE STRING STORAGE.
129  REM WHEN LATER RECALLED WITH
     THE ADDED OFFSET (FIX), IT
     WOULD POINT ELSEWHERE IN THE
     PROGRAM.
130 A$(0) = "HI," + " ": REM  CAL
     CULATION
135  INPUT "YOUR NAME PLEASE ";A$
     (1): REM  INPUTTING
140 A$(100) = "  BY THE WAY. " +
     " "
150 B$(1,1) = "MY" + " "
160 B$(50,2) = "CASSETTE" + " "
170 C$(1,1,1) = "LOAD IS" + " "
180 C$(70,2,1) = "FASTER THAN" +
     " "
190 C$(85,0,0) = "DISK" + " "
200 C$(100,1,0) = "LOAD !!!" + "!
     "
300  REM  SAVE TO TAPE
310 Z =  FRE (0): REM TO CONDENSE
      STRING STORAGE
320  PRINT "PRESS RETURN WHEN REA
     DY": GET Z$: STORE A$: STORE
     B$: STORE C$: CALL 768
400  END
30000  DATA 160,111,162,0,132,60,
     134,61,160,116,32,25,3,164,1
     11,166,112,132,60,134,61,164
     ,115,166,116,132,62,134,63,7
     6,205,254,160,24,162,0,132,6
     0,134,61,160,29,132,62,134,6
     3,32,253,254,165,28,56,229,2
     4,133,8,165
30010  DATA 29,229,25,133,9,160,2
     ,162,0,56,181,111,149,62,245
     ,8,149,111,149,60,232,136,20
     8,242,165,24,56,229,111,133,
     6,165,25,229,112,133,7,76,25
     3,254,32,190,222,32,123,221,
     32,108,221,160,2,165,155,24,
     113,155,133
30020  DATA 8,200,165,156,113,155
     ,133,9,160,0,177,160,240,15,
     200,177,160,56,229,6,145,160
     ,200,177,160,229,7,145,160,1
     69,3,24,101,160,133,160,144,
     2,230,161,197,8,208,220,165,
     9,197,161,208,214,96
```

Listing 2

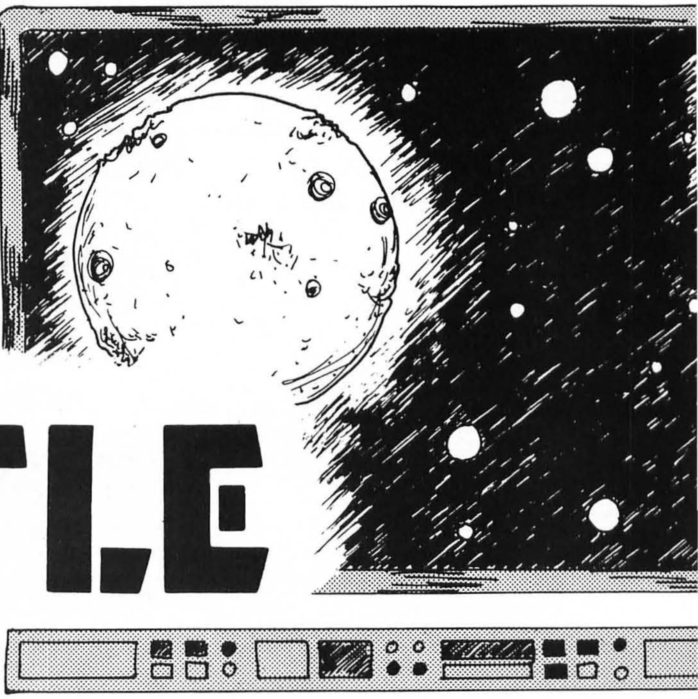
```
50  FOR Z = 768 TO 932: READ A: POKE
     Z,A: NEXT
100  DIM A$(100)
110  DIM B$(100,2)
120  DIM C$(100,2,1)
125  GOSUB 300
130  PRINT A$(0);
135  PRINT A$(1)
140  PRINT A$(100);
150  PRINT B$(1,1);
160  PRINT B$(50,2);
170  PRINT C$(1,1,1);
180  PRINT C$(70,2,1);
190  PRINT C$(85,0,0);
200  PRINT C$(100,1,0);: END
300  REM   LOAD FROM TAPE
310 Z =  FRE (0): REM TO CONDENSE
      STRING STORAGE
320  PRINT "PRESS RETURN WHEN REA
     DY": GET Z$: RECALL A$: RECALL
     B$: RECALL C$: CALL 800: CALL
     865,C$(0,0,0): CALL 865,A$(0
     ): CALL 865,B$(0,0)
330  RETURN
400  END
30000  DATA 160,111,162,0,132,60,
     134,61,160,116,32,25,3,164,1
     11,166,112,132,60,134,61,164
     ,115,166,116,132,62,134,63,7
     6,205,254,160,24,162,0,132,6
     0,134,61,160,29,132,62,134,6
     3,32,253,254,165,28,56,229,2
     4,133,8,165
30010  DATA 29,229,25,133,9,160,2
     ,162,0,56,181,111,149,62,245
     ,8,149,111,149,60,232,136,20
     8,242,165,24,56,229,111,133,
     6,165,25,229,112,133,7,76,25
     3,254,32,190,222,32,123,221,
     32,108,221,160,2,165,155,24,
     113,155,133
30020  DATA 8,200,165,156,113,155
     ,133,9,160,0,177,160,240,15,
     200,177,160,56,229,6,145,160
     ,200,177,160,229,7,145,160,1
     69,3,24,101,160,133,160,144,
     2,230,161,197,8,208,220,165,
     9,197,161,208,214,96
```

Listing 3

```
45 WTAPE = 768:RETAPE = 800:FIX =
     865
50  FOR Z = 768 TO 932: READ A: POKE
     Z,A: NEXT : FOR Z = 1 TO 200
     0: NEXT Z: GOTO 100
4190 Z =  FRE (0): INPUT "START R
     ECORDER AND PRESS RETURN ";X
     $
4215  CALL WTAPE
5210  FOR Z = 0 TO 1000:L$(Z) = "
     ": NEXT :Z =  FRE (0): INPUT
     "START RECORDER AND PRESS RE
     TURN ";X$
5235  CALL RETAPE
5238  CALL FIX,L$(0)
5239 Z =  FRE (0)
20010  IF E = 0 OR E > 15 THEN  TEXT
     : PRINT "ERROR #";E;" IN LIN
     E "; PEEK (218) +  PEEK (219
     ) * 256: STOP
30000  DATA 160,111,162,0,132,60,
     134,61,160,116,32,25,3,164,1
     11,166,112,132,60,134,61,164
     ,115,166,116,132,62,134,63,7
     6,205,254,160,24,162,0,132,6
     0,134,61,160,29,132,62,134,6
     3,32,253,254,165,28,56,229,2
     4,133,8,165
30010  DATA 29,229,25,133,9,160,2
     ,162,0,56,181,111,149,62,245
     ,8,149,111,149,60,232,136,20
     8,242,165,24,56,229,111,133,
     6,165,25,229,112,133,7,76,25
     3,254,32,190,222,32,123,221,
     32,108,221,160,2,165,155,24,
     113,155,133
30020  DATA 8,200,165,156,113,155
     ,133,9,160,0,177,160,240,15,
     200,177,160,56,229,6,145,160
     ,200,177,160,229,7,145,160,1
     69,3,24,101,160,133,160,144,
     2,230,161,197,8,208,220,165,
     9,197,161,208,214,96
```

SoftSide **ATARI® DV**

ROBOT BATTLE



by Randy W. Massey

***Robot Battle* is an arcade-type game for an ATARI® with 32K RAM, one joystick, BASIC language cartridge, and one 810 disk drive. It is included as a bonus program on this month's ATARI® DV.**

An alien race, the Organians, have declared war against Earth. They have penetrated the Earth's Orbiting Space Station and have stolen the cloaking device. Without this device, the Earth is defenseless against enemy invasion. Your orders are to penetrate the Organian spaceship, retrieve the cloaking device, and return to Earth.

The alien spacecraft has five rooms and you must pass through each in order to get to the cloaking device. The device itself is surrounded by a pulsating energy field. Contact with the field is instantly fatal. Numerous barriers are positioned throughout the alien ship and running into one may cause you to lose one of your moves. However, you may find the barriers useful in avoiding the aliens' laser fire.

Within the alien ship are marauding robots, programmed to destroy any intruders. There is also a strange spider-like creature that, when wounded, mutates into an extremely dangerous and aggressive form. This new creature possesses an unknown incredible power source that can destroy the surrounding robots or any barriers in their way. Avoid it if you can.

To play *Robot Battle*, plug a joystick into slot number one and type from BASIC, RUN"D: ROBOT". Or simply choose the program ROBOT as listed in the ATARI® disk cover program. After *Robot Battle* loads into computer memory, the title screen is displayed and you then enter your game options.

Before beginning to play, the computer asks you to select a skill level. Novice is the easiest, having four moves per turn. There are fewer attacking aliens and your shields are able to absorb more of the enemy laser fire. Intermediate allows three turns per move and has more aliens and weaker shields.

The expert level is only for persons who have played this game a great many times. You're allowed two moves per turn, and the number of aliens and the strength of their weapons are formidable. Anyone who can succeed at this level is a master game player.

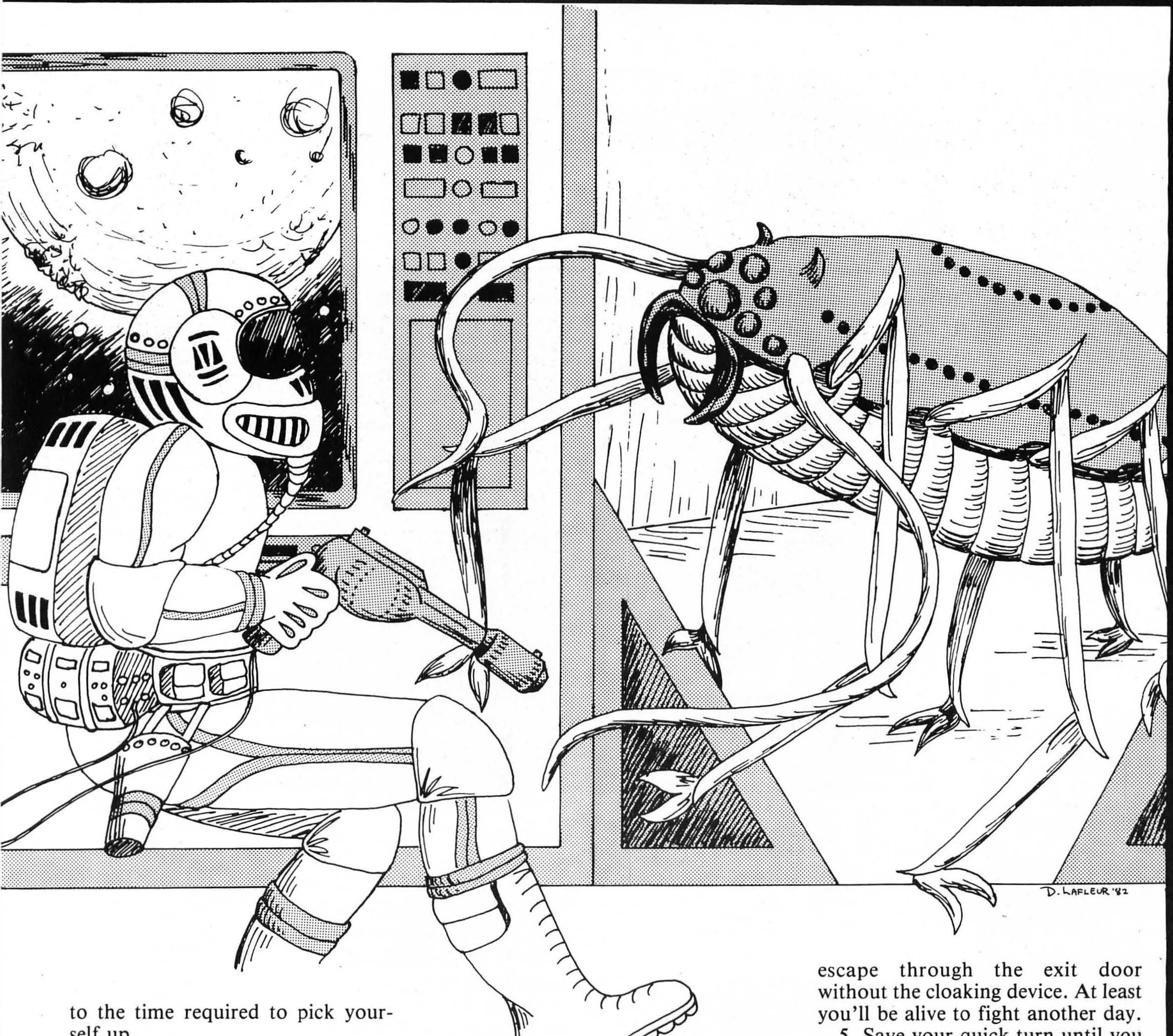
After making your selection, press the START key to begin. During the game, prompts are given to inform you of your next move. The game starts with the player's move first. You are limited to ten seconds to complete your turn.

Your weapon consists of a high-power laser that is capable of incinerating anything in its path. To fire it, simply press the joystick button, then aim the joystick in the direction you want to fire. Each firing of your laser counts as one turn. Should you be fired upon, your shields will be reduced from the initial amount of 100% to a lower value based on the distance of the alien and the skill level you selected. Hits from close up are particularly dangerous.

Destroying spiders always takes at least two shots because their hard chitinous shells are able to absorb several hits. When wounded, a spider will shed its shell and metamorphose into an octopus-like creature. This creature can fire up to three shots and will even destroy the surrounding robots should they get in its way.

The robots are easier to destroy than the spiders, but some will take more than one shot. If damaged, they change color and lose the ability to move; however, they can still fire.

Quick thinking is needed to determine the best path to the exit door which is located on the perimeter of the room. To leave a room, simply place yourself at this door. Barriers are located throughout the rooms. If you accidentally bump into one, you may lose one of your moves due



to the time required to pick yourself up.

Your display at the bottom of the screen consists of four items of information. The most important of these is about your shields, since if you lose your shields you will certainly be destroyed. Also shown is the current room number, the number of robots, and finally the number of spiders.

During the game you have one chance to escape from a dangerous situation. To use the PANIC BUTTON, press any key and you can take as many as ten quick moves. You must be quick, as you are still limited to ten seconds.

Suggested Strategy

1. Use the barriers to your advantage. Remember that the robots cannot destroy them, but the spiders can.

2. Put the wounded spiders to work for you. When wounded, they destroy some of the other attacking aliens, making your job easier.

3. Sometimes an alien will place itself between you and another alien. This will effectively block the first alien's shot. The more aliens on either side of the line of fire, the greater the chance that shots will be blocked by another alien who happens to move into the line of fire.

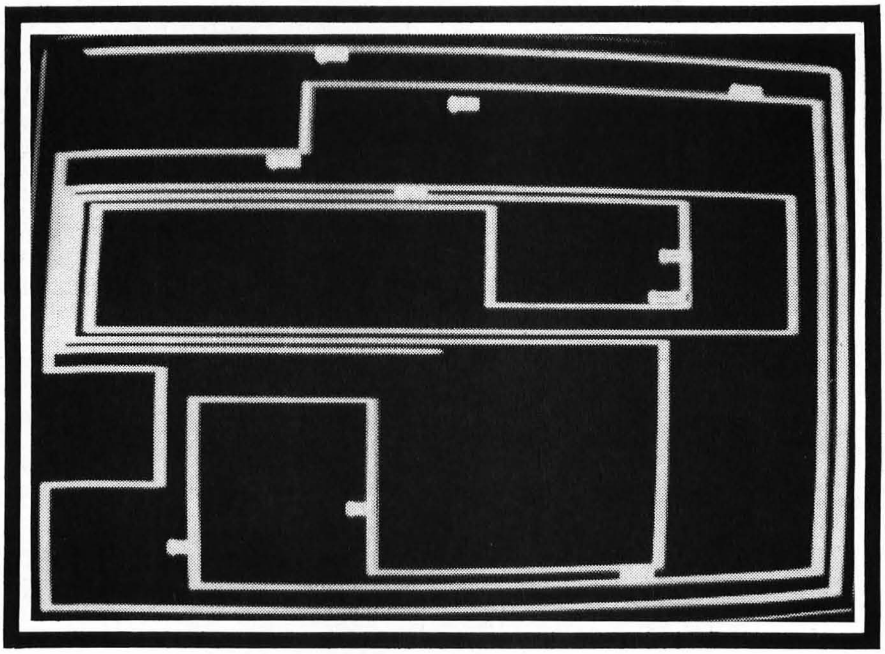
4. If you make it to the fifth room and your shields are low, you can escape through the exit door without the cloaking device. At least you'll be alive to fight another day.

5. Save your quick turn until you really need it. To get the most distance, take one turn less than the maximum number allowed (e.g., three in the novice level) before you press a key. This will give you a total of 13 moves in only one turn. Those extra three moves may make all the difference.

Programs on the Disk

ROBOT: Loads display list interrupt and joystick Machine Language routines into page 6 and bumps lomem up 2K to make room for two character sets, then loads main program.
ROBOT.PT2: Main program.
ROBOT.PT3: First character set.
ROBOT.PT4: Second character set.

K-Bytes



Super-Snake

An ATARI® K-Byter by Trevor Porter, Sarasota, FL

The object of this game is to hit as many of the white boxes as possible with your "snake." The game ends as soon as you run into a wall or your own trail. To move, you can use either a joystick or the arrow keys on the keyboard. If you find yourself in a tight spot, you can either press the FIRE button or the RETURN key to jump to a new location on the screen.

Scoring is based on how many boxes are hit and how long the game is played. More points are scored for hitting the edge of a box than for going through its middle.

```
10 GRAPHICS 9:POKE 710,0:ZX=1:ZY=0:X=4
0:Y=90:COLOR 10:PLOT 0,0:DRAWTO 79,0:D
RAWTO 79,191:DRAWTO 0,191:DRAWTO 0,0
20 A=RND(0)*70+5:B=RND(0)*180+5:COLOR
15:PLOT A,B:DRAWTO A+2,B:DRAWTO A+2,B+
2:DRAWTO A,B+2:DRAWTO A,B
30 COLOR 5:LOCATE X,Y,Z:IF Z=10 OR Z=5
 THEN 100
40 PLOT X,Y:IF Z=15 THEN 110
50 S=STICK(0):P=PEEK(764):IF S=14 OR P
=14 THEN ZX=0:ZY=-1
60 IF S=13 OR P=15 THEN ZX=0:ZY=1
70 IF S=11 OR P=6 THEN ZX=-1:ZY=0
80 IF S=7 OR P=7 THEN ZX=1:ZY=0
85 IF P=12 OR STRIG(0)=0 THEN POKE 764
,255:X=RND(0)*70+5:Y=RND(0)*180+5
90 X=X+ZX:Y=Y+ZY:SC=SC+1:GOTO 30
100 GRAPHICS 0:? "SCORE - ";SC:CLR :FO
R I=0 TO 999:NEXT I:RUN
110 X=X+ZX:Y=Y+ZY:SC=SC+1000:GOTO 20
```

CROSS REFERENCE

by Luis Wuhl

Cross Reference **is an ATARI® program requiring a 24K disk system with optional printer.**

One of the more important parts in writing a program is the debugging phase. If your program isn't totally bug-free the chances of getting useful results are slim.

To aid in the debugging process, several utilities are commercially available to the programmer. Some of the more valuable utilities are a program compressor, line renumberer, line cross reference, variable cross reference, and a program trace/monitor.

The two most important of these are the renumberer and the variable cross reference. (Last month's ATARI® DV featured a renumber utility.) A good variable reference utility should display a listing of all variables and the line numbers in which they appear. This program qualifies in both respects.

To use *Cross Reference*, first load in the program that will be searched. When the ATARI® responds "READY", be sure the diskette has enough free space on it and type:

```
LIST"D:filespec.ext"
```

Where "filespec" is any eight-letter file name and "ext" is a three-letter extension.

When the ATARI® again says "READY", load in this cross reference program and type RUN. Answer the question now displayed by typing in the "D:filespec.ext" that you used before. The disk will start spinning and will turn on and off from time to time.

The time it takes to reference a program depends on its size. A short program can take up to a minute whereas a very large program may take up to 20 or 30 minutes.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$        ATARI  BASIC        $
$  'ATARI Cross Reference'   $
$      Author: Luis Wuhl     $
$     (c) 1982, SoftSide     $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

10 DIM REG$(140),POS$(100),VAR$(5120),
AZ$(20):A=FRE(0)*0.8:DIM ST$(A),PROG$(
30):OPEN #2,4,0,"K:"
20 GOSUB 7000
22 TOTVAR=0:CUEN=1
35 GOSUB 2000:REM READ
37 IF REG$="FIN" THEN GOSUB 15000:END
40 GOSUB 1000:REM LOOK FOR BLANKS
50 LNUM=VAL(REG$(1,N-1))
60 LAST=N+1
70 GOSUB 1000:REM LOOK FOR TERMINATOR
75 IF FIN THEN 35:REM END INSTRUCTION
77 IF LAST>N-1 THEN 60
80 POS$=REG$(LAST,N-1)
84 IF POS$="DATA" THEN 35
85 IF POS$="REM" THEN 35
86 GOSUB 5000:REM CHECK FOR A NUMBER
87 IF NUM THEN 60
110 GOSUB 4000:REM CHECK INSTRUCTION
120 IF NO THEN 60
130 GOSUB 13000
140 GOTO 60
999 REM SEARCH FOR BLANKS
1000 IF N>=LEN(REG$) THEN FIN=1:GOTO 1
020
1007 N=N+1:AS=ASC(REG$(N,N))
1008 IF ASC(REG$(LAST))=34 THEN GOSUB
3040:GOTO 1000
1010 IF  NOT (AS>=48 AND AS<=90) THEN
1015
1013 FOR G=58 TO 64:IF AS<>G THEN NEXT
 G:GOTO 1000
1014 POP
1015 IF REG$(N,N)="?" THEN 1000
1020 RETURN
1999 REM INPUT PROGRAM LINE
2000 TRAP 2050
2012 FIN=0:N=0:LAST=1
2020 INPUT #1,REG$
2022 REG$(LEN(REG$)+1)=" "
2030 SENT=SENT+1
2040 TRAP 40000:RETURN
2050 REG$="FIN":RETURN
2999 REM CHECK FOR COMMAS
3000 NO=1
3010 FOR H=1 TO LEN(POS$)
3020 IF ASC(POS$(H,H))=34 THEN 3040
3030 NEXT H:NO=0:RETURN
3039 REM CHECK FOR COMMAS
3040 N=N+1
3050 N=N+1:IF ASC(REG$(N,N))<>34 THEN
3050
3060 N=N+1:LAST=N+1:RETURN
3999 REM CHECK FOR INSTRUCTION
4000 NO=1
4100 IF POS$="ABS" THEN RETURN
4101 IF POS$="ADR" THEN RETURN
4102 IF POS$="AND" THEN RETURN
4103 IF POS$="ASC" THEN RETURN
4104 IF POS$="ATN" THEN RETURN
4105 IF POS$="BYE" THEN RETURN
4106 IF POS$="CLOAD" THEN RETURN
4107 IF POS$="CHR$" THEN RETURN
4108 IF POS$="CLOG" THEN RETURN
4109 IF POS$="CLOSE" THEN RETURN
4110 IF POS$="CLR" THEN RETURN
4111 IF POS$="COLOR" THEN RETURN
4112 IF POS$="COM" THEN RETURN
4113 IF POS$="CONT" THEN RETURN
4114 IF POS$="COS" THEN RETURN
4115 IF POS$="CSAVE" THEN RETURN
4116 IF POS$="DEG" THEN RETURN
4117 IF POS$="DIM" THEN RETURN
4118 IF POS$="DOS" THEN RETURN
4119 IF POS$="DRAWTO" THEN RETURN
4120 IF POS$="END" THEN RETURN
4121 IF POS$="ENTER" THEN RETURN
4122 IF POS$="EXP" THEN RETURN
4123 IF POS$="FOR" THEN RETURN
4124 IF POS$="FRE" THEN RETURN
4125 IF POS$="GET" THEN RETURN
4126 IF POS$="GOSUB" THEN RETURN
4127 IF POS$="GOTO" THEN RETURN
4128 IF POS$="GO TO" THEN RETURN
4129 IF POS$="GRAPHICS" THEN RETURN
4130 IF POS$="IF" THEN RETURN
4131 IF POS$="INPUT" THEN RETURN
4132 IF POS$="INT" THEN RETURN
4133 IF POS$="LEN" THEN RETURN
4134 IF POS$="LET" THEN RETURN
4135 IF POS$="LIST" THEN RETURN
4136 IF POS$="LOAD" THEN RETURN
4137 IF POS$="LOCATE" THEN RETURN
4138 IF POS$="LOG" THEN RETURN
4139 IF POS$="LPRINT" THEN RETURN
4140 IF POS$="NEW" THEN RETURN
4141 IF POS$="NEXT" THEN RETURN
4142 IF POS$="NOT" THEN RETURN
4143 IF POS$="NOTE" THEN RETURN
4144 IF POS$="ON" THEN RETURN
4145 IF POS$="OPEN" THEN RETURN
4146 IF POS$="OR" THEN RETURN
4147 IF POS$="PADDLE" THEN RETURN
4148 IF POS$="PEEK" THEN RETURN
4149 IF POS$="PLOT" THEN RETURN
4150 IF POS$="POINT" THEN RETURN
4151 IF POS$="POKE" THEN RETURN
4152 IF POS$="POP" THEN RETURN
4153 IF POS$="POSITION" THEN RETURN
4154 IF POS$="PRINT" THEN RETURN
4155 IF POS$="PTRIG" THEN RETURN
4156 IF POS$="PUT" THEN RETURN
4157 IF POS$="RAD" THEN RETURN
4158 IF POS$="READ" THEN RETURN
4159 IF POS$="RESTORE" THEN RETURN
4160 IF POS$="RETURN" THEN RETURN
4161 IF POS$="RND" THEN RETURN
4162 IF POS$="RUN" THEN RETURN
4163 IF POS$="SAVE" THEN RETURN
4164 IF POS$="SETCOLOR" THEN RETURN
4165 IF POS$="SGN" THEN RETURN
4166 IF POS$="SIN" THEN RETURN
4167 IF POS$="SOUND" THEN RETURN
4168 IF POS$="SQR" THEN RETURN
4169 IF POS$="STATUS" THEN RETURN
4170 IF POS$="STEP" THEN RETURN
4171 IF POS$="STICK" THEN RETURN
4172 IF POS$="STRIG" THEN RETURN
4173 IF POS$="STOP" THEN RETURN
4174 IF POS$="STR$" THEN RETURN
4175 IF POS$="THEN" THEN RETURN
4176 IF POS$="TO" THEN RETURN
4177 IF POS$="TRAP" THEN RETURN
4178 IF POS$="USR" THEN RETURN
4179 IF POS$="VAL" THEN RETURN
4180 IF POS$="XIO" THEN RETURN
4190 IF POS$="?" THEN RETURN
4200 NO=0:RETURN
4999 REM CHECK FOR A NUMBER
5000 NUM=1
5020 IF POS$(1,1)<"0" OR POS$(1,1)>"9"
 THEN NUM=0
5030 RETURN
6999 REM INPUT PROGRAM NAME
7000 ? "}"
7020 POSITION 2,10:? "ENTER PROGRAM NA
ME: (Dn:Filespec.Ext)"
7030 INPUT PROG$
7040 TRAP 7200
7050 OPEN #1,4,0,PROG$
7060 TRAP 40000
7080 RETURN
7200 CLOSE #1:GOTO 7000
7999 REM CHECK FOR STRING OR ARRAY
8000 IF N=LEN(REG$) THEN 8100
8020 IF REG$(N,N)="(" OR REG$(N,N)="$"
 THEN POS$(LEN(POS$)+1)=REG$(N,N)
8100 RETURN
```

```
12999 REM STORE VARIABLES
13000 GOSUB 8000
13010 GOSUB 14000
13015 ST$(CUEN,CUEN+5)="      "
13020 ST$(CUEN,CUEN)=CHR$(NVAR):ST$(CU
EN+1,CUEN+5)=STR$(LNUM):CUEN=CUEN+6
13040 RETURN
13999 REM CHECK REPEAT OF VARIABLE
14000 IF LEN(POS$)<40 THEN POS$(LEN(PO
S$)+1)=" ":GOTO 14000
14040 NVAR=0
14045 IF LEN(VAR$)=0 THEN 14090
14050 FOR T=1 TO LEN(VAR$) STEP 40
14060 NVAR=NVAR+1
14070 IF POS$=VAR$(T,T+39) THEN POP :G
OTO 14100
14080 NEXT T
14090 TOTVAR=TOTVAR+1:NVAR=NVAR+1:VAR$
(LEN(VAR$)+1)=POS$
14100 RETURN
14999 REM PRINT VARIABLS AND LINE #'S
15000 POKE 752,1:? "OUTPUT TO WHICH DE
VICE? (E/P/R)":GET #2,AZ:AZ$=CHR$(AZ)
15002 IF AZ$="P" THEN TRAP 15100:LPRIN
T " "
15003 OPEN #5,8,0,AZ$:TRAP 40000
15004 ? #5;"CROSS REFERENCES OF ";PROG
$:? #5;"=================================
=====":? #5;
15005 NVAR=0
15010 FOR T=1 TO LEN(VAR$) STEP 40
15020 NVAR=NVAR+1
15025 ? #5;VAR$(T,T+39);":"
15030 FOR G=1 TO LEN(ST$) STEP 6
15040 IF ASC(ST$(G,G))=NVAR THEN ? #5;
VAL(ST$(G+1,G+5));", ";
15050 NEXT G:? #5;"DONE"
15055 ? #5;"----------------------------
---------"
15060 NEXT T:? #5
15065 ? #5;" NUMBER OF PROGRAM LINES "
;SENT
15067 ? #5:? #5;" NUMBER OF VARIABLES
DEFINED ";TOTVAR
15070 CLOSE #5:RETURN
15100 POSITION 3,6:? "}PRINTER NOT REA
DY}"
15110 POSITION 3,7:? "PRESS RETURN WHE
N READY ";:INPUT POS$:GOTO 15000
```

by Skeet Nevil

***Word Search* is a word game program for a 16K ATARI® with one joystick.**

In this version of word search, words are randomly written up, down, forward, backward, or on a diagonal. Two different words may share the same letter. At levels 1 and 3, five words are randomly selected from a list of fifty easy words and placed in a 10 by 10 square. At levels 2 and 4, ten words are selected from a list of fifty computer-related words and placed in a 20 by 20 square. The remaining positions in the square are filled with randomly selected letters. Levels 3 and 4 are the same as 1 and 2 respectively, except that the letters are invisible.

One of the hidden words will be displayed at the bottom of the screen. Use the joystick (plugged into controller slot #1) to find the first letter of the word. Then press the trigger. If you are right, the word will be highlighted and the next one displayed. A warning message will be displayed if you are wrong. Three misses will terminate the game.

Press the space bar to view the next word at the bottom of the screen. Only words not found yet will be displayed. The list will start over again after viewing all words. When you find a word, make sure it is displayed at the bottom before pressing the trigger. Failure to do so will result in a miss.

Press "C" to get a clue for a word. The clue will tell the direction in which the word is written. For example, "NW TO SE" means that the word starts in the northwest direction and is written toward the southeast. To give up on a word, press "G". The word will then be displayed.

At the bottom, "REMAINING" is the number of words left, "VALUE" is the point value of the current word, and "SCORE" is your current score. Words are worth twenty points for the first minute of the game and decrease in value by two points every minute. After nine minutes, the value will remain at two points. Requesting a clue will cut the point value in half for that word. No points will be awarded for a word that was given up.

Variables

C(10): Stores "1" if a clue was requested for that word.
D(10): Stores the direction of each word.
L(10): Stores the length of each word.
M: The number of misses.
MN: Minimum X and Y values.
MX: Maximum X and Y values.
NW: Number of words.
R: Number of words remaining.
S: The score.
T(19,19): Stores the ATASCII codes of the letters of each selected word.
V: Value of the current word.
W$: Contains all the selected words. Twelve bytes are reserved for each word. The end of a word is denoted by "*".
X(10),Y(10): Store the starting X and Y coordinates of each selected word.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$          Atari BASIC         $
$         'WORD SEARCH'        $
$      Author: Skeet Nevil     $
$      (c) 1982,  SoftSide     $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
10 GOTO 500
```

Subroutine to find the change in X and Y depending on the direction (D) of the word.

```
20 GOTO 20+D
21 X1=1:Y1=0:GOTO 30
22 X1=1:Y1=1:GOTO 30
23 X1=0:Y1=1:GOTO 30
24 X1=-1:Y1=1:GOTO 30
25 X1=-1:Y1=0:GOTO 30
26 X1=-1:Y1=-1:GOTO 30
27 X1=0:Y1=-1:GOTO 30
28 X1=1:Y1=-1:GOTO 30
30 RETURN
```

Subroutine to determine if the word can be displayed. If so, it is stored in the T table.

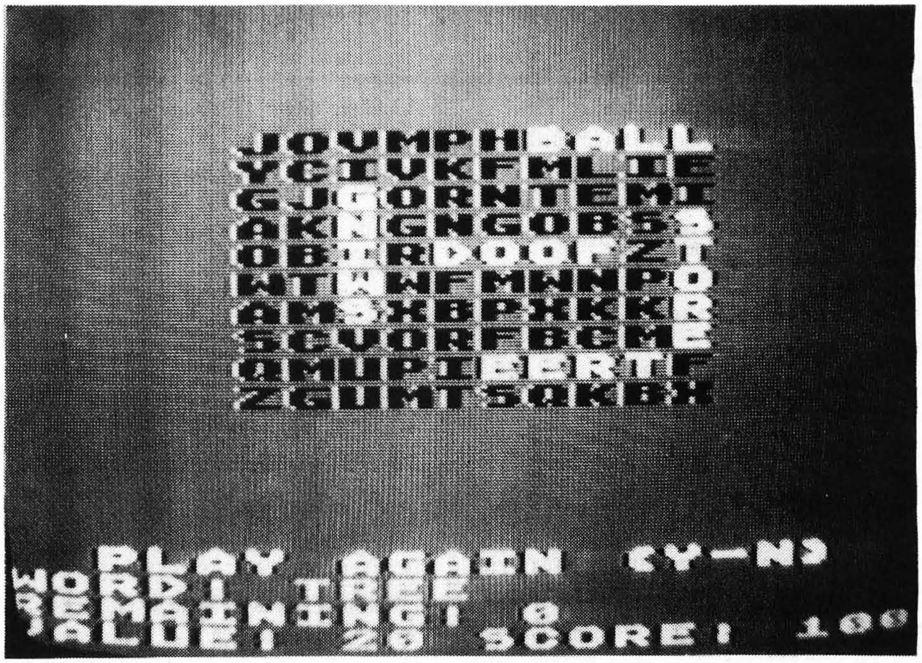
```
40 X2=X:Y2=Y:E=0
50 FOR J=1 TO LEN(T$)
60 IF X2<MN OR X2>MX OR Y2<MN OR Y2>MX
 THEN E=1:POP :GOTO 120
70 IF T(X2,Y2)=0 THEN 80
75 IF ASC(T$(J,J))<>T(X2,Y2) THEN E=1:
POP :GOTO 120
80 X2=X2+X1:Y2=Y2+Y1:NEXT J
100 FOR J=1 TO LEN(T$):T(X,Y)=ASC(T$(J
,J))
110 X=X+X1:Y=Y+Y1:NEXT J
120 RETURN
```

Subroutine to place the chosen words on the screen and fill the rest of the screen with random letters.

```
130 FOR Y=MN TO MX:P=100:FOR X=MN TO M
X:SOUND 0,P,10,10:P=P-4
140 POSITION X,Y:IF T(X,Y)<>0 THEN PRI
NT #6;CHR$(T(X,Y)):GOTO 160
150 PRINT #6;CHR$(INT(RND(0)*26)+65)
160 NEXT X:NEXT Y
170 SOUND 0,0,0,0:RETURN
```

Subroutine to print the number of remaining words (180), value of a word (190), and score (200).

```
180 POSITION 11,22:T$=STR$(R):GOTO 210
190 POSITION 7,23:T$=STR$(V):GOTO 210
```



```
JOUMPHBALL
YCIUKFMLIE
GJGORNTEMI
AKNGNGOBSS
OBIRDOOFZT
WTWWFMWNPO
AMSXBPXKKR
SCVORFBCME
QMUPIEERTF
ZGUMTSQKBX

  PLAY AGAIN  (Y-N)
WORD: TREE
REMAINING: 0
VALUE: 20   SCORE: 100
```

```
200 POSITION 17,23:T$=STR$(S):GOTO 210
210 FOR I=1 TO LEN(T$):PRINT #6;CHR$(A
SC(T$(I,I))+96);:NEXT I:IF I=2 THEN PR
INT #6;" "
220 RETURN
```

Subroutine to print the word that starts at X(W), Y(W), in direction D(W) for a length of L(W).

```
230 X2=X(W):Y2=Y(W):D=D(W):GOSUB 20
240 FOR I=1 TO L(W)
242 IF T(X2,Y2)>192 THEN 255
243 IF T(X2,Y2)>96 AND A=128 THEN A1=9
6:GOTO 248
246 IF T(X2,Y2)>96 AND A=32 THEN 255
247 A1=A
248 T(X2,Y2)=T(X2,Y2)+A1
249 IF X=X2 AND Y=Y2 THEN C=C+A1:GOTO
255
250 POSITION X2,Y2:PRINT #6;CHR$(T(X2,
Y2))
255 X2=X2+X1:Y2=Y2+Y1
260 NEXT I
270 P=12*W-11:W$(P,P)="*"
280 R=R-1:GOSUB 180
290 RETURN
```

Subroutine to print the clue.

```
300 POSITION 3,20:PRINT #6;"CLUE:";
310 ON D(W) GOTO 311,312,313,314,315,3
16,317,318
311 T$="W TO E":GOTO 320
312 T$="NW TO SE":GOTO 320
313 T$="N TO S":GOTO 320
314 T$="NE TO SW":GOTO 320
315 T$="E TO W":GOTO 320
316 T$="SE TO NW":GOTO 320
317 T$="S TO N":GOTO 320
318 T$="SW TO NE":GOTO 320
320 PRINT #6;T$
330 C(W)=1:V=V/2:GOSUB 190
340 FOR I=1 TO 6:FOR P=121 TO 96 STEP
-1:SOUND 0,P,10,10:NEXT P
345 FOR P=96 TO 121:SOUND 0,P,10,10:NE
XT P:NEXT I:SOUND 0,0,0,0
350 POSITION 3,20:PRINT #6;"
   "
360 RETURN
```

Subroutine to play the word-found tune.

```
370 RESTORE 430:FOR I=1 TO 6:READ P,D:
FOR J=1 TO D:SOUND 0,P,10,10:NEXT J:NE
XT I:SOUND 0,0,0,0:RETURN
```

Subroutine to make the miss sound.

```
380 LO=57:HI=45:NT=HI:FOR T=0 TO 6:SOU
ND 0,NT,10,14
381 FOR I=1 TO 120:NEXT I
382 NT=LO:LO=HI:HI=NT:NEXT T
383 SOUND 0,0,0,0:RETURN
```

Subroutine for the press-key sound.

```
390 FOR I=1 TO 30:SOUND 0,29,10,10:NEX
T I:SOUND 0,0,0,0:RETURN
```

Subroutine for the give-up sound.

```
395 FOR P=60 TO 121:SOUND 0,P,10,10:NE
XT P:SOUND 0,0,0,0:RETURN
```

Data for the title screen.

```
400 DATA JXHHVAPLMMOK,CwordKIOBPCR,FLT
YPDsWTRZM,NYNNXPOeKNWI,GIRULPADaHLS,NW
IUYSCZQrPU
401 DATA BHXSRCHRHFcL,BDybZMYXJHTh,MZK
MDFGRWTLP,DHDUskeetBXE,PMOHVOnevilA,NI
WOPAJNUCDD
```

Words for levels 1 and 3.

```
410 DATA TREE*,HOUSE*,BOAT*,BOY*,GIRL*
,BIG*,STORE*,PRETTY*,HELP*,FOOD*
411 DATA BROWN*,HIDE*,WATER*,CAT*,APPL
E*,GRASS*,SKY*,BOOK*,DISH*,FAST*
412 DATA PLAY*,CUP*,EAT*,RADIO*,GLAD*,
STONE*,BIKE*,TOY*,SWING*,RIVER*
413 DATA BALL*,FORK*,SPOON*,SAID*,FLOW
ER*,BEE*,LEAVE*,SORRY*,PLEASE*,THANK*
414 DATA BIRTHDAY*,HOME*,CAR*,WAGON*,M
ILK*,COOKIE*,DESERT*,SAND*,MAYBE*,YES*
```

Words for levels 2 and 4.

```
420 DATA ATARI*,COMPUTER*,BYTE*,DEBUG*
,FLOWCHART*,BASIC*,PROGRAM*,SUBROUTINE
*,ADDRESS*,MEMORY*
421 DATA BINARY*,HEXADECIMAL*,LOGIC*,C
OBOL*,FORTRAN*,DATA*,ASSEMBLER*,COMPIL
E*,ERROR*,JUMP*
422 DATA GOTO*,CURSOR*,DISPLAY*,DISK*,
TAPE*,ARITHMETIC*,FILE*,FUNCTION*,LIBR
ARY*,PEEK*
423 DATA POKE*,GOSUB*,RETURN*,GRAPHICS
*,INPUT*,OUTPUT*,KEYBOARD*,LOOP*,MATRI
X*,STRING*
424 DATA VARIABLE*,RANDOM*,PIXEL*,COOR
DINATE*,STACK*,PRINTER*,RESTORE*,MODE*
,POSITION*,EXECUTE*
```

Data for the word-found tune.

```
430 DATA 121,15,96,15,81,15,60,40,81,1
5,60,40
```

Initialization.

```
500 DIM W$(120),T$(12),T(19,19),X(10),
Y(10),D(10),L(10),C(10)
510 GRAPHICS 17:SETCOLOR 0,0,4:SETCOLO
R 1,0,4:SETCOLOR 4,0,4
```

Display the title screen.

```
520 FOR I=4 TO 15:READ T$:POSITION 4,I
:PRINT #6;T$:NEXT I
530 GOSUB 390:SETCOLOR 0,0,0:SETCOLOR
1,0,0:FOR I=1 TO 500:NEXT I:GOSUB 390:
SETCOLOR 1,12,10
540 POSITION 1,22:PRINT #6;"instructio
ns ";CHR$(8);"y";CHR$(13);"n";CHR$(9)
```

Determine if instructions are requested or not.

```
550 POKE 764,255
560 IF PEEK(764)=255 THEN 560
570 KEY=PEEK(764):IF KEY=35 THEN GOSUB
 390:GOTO 710
580 IF KEY<>43 THEN 550
585 GOSUB 390
```

Print instructions.

```
590 GRAPHICS 0:PRINT "    Words will b
e randomly placed on":PRINT "the scree
n. They may be spelled fore-"
600 PRINT "ward, backwards, up, down,
or on a":PRINT "diagonal.":PRINT :PRIN
T "    Use the joystick to find the"
610 PRINT "first letter of a word and
press the":PRINT "trigger. You will be
 allowed only two"
620 PRINT "misses. The game will end o
n the":PRINT "third miss.":PRINT
630 PRINT "    Use the SPACE BAR to li
st the":PRINT "hidden words. Press 'C'
 for a clue"
640 PRINT "or 'G' to give up on a word
.":PRINT :PRINT "    Words are worth 2
0 points for the"
650 PRINT "first minute and decrease i
n value by":PRINT "2 points for each a
dditional minute."
660 PRINT "After 9 minutes, words will
 remain at":PRINT "2 points each. A cl
ue will cut the"
670 PRINT "point value in half for tha
t word. A":PRINT "word that was given
up is worth zero."
680 PRINT :PRINT "    Press any key t
o continue.";
```

Continue if any key is pressed.

```
690 POKE 764,255
700 IF PEEK(764)=255 THEN 700
705 GOSUB 390
```

Print screen to allow choice of level.

```
710 GRAPHICS 17:SETCOLOR 0,12,10:SETCO
LOR 4,0,4
720 POSITION 0,2:PRINT #6;"1: FIVE EAS
Y WORDS":POSITION 3,3:PRINT #6;"100 LE
TTERS"
725 POSITION 0,5:PRINT #6;"2: TEN COMP
UTER":POSITION 3,6:PRINT #6;"RELATED W
ORDS"
730 POSITION 3,7:PRINT #6;"400 LETTERS
":POSITION 0,9:PRINT #6;"3: SAME AS 1
EXCEPT"
735 POSITION 3,10:PRINT #6;"INVISIBLE
LETTERS":POSITION 0,12:PRINT #6;"4: SA
ME AS 2 EXCEPT"
740 POSITION 3,13:PRINT #6;"INVISIBLE
LETTERS":POSITION 5,16:PRINT #6;"LEVEL
 (1-4)"
750 POKE 764,255
```

Determine the level.

```
760 KEY=PEEK(764):IF KEY=31 OR KEY=26
THEN MX=14:MN=5:NW=5:GOTO 790
770 IF KEY=30 OR KEY=24 THEN MX=19:MN=
0:NW=10:GOTO 790
780 GOTO 760
790 GOSUB 390:IF KEY=31 OR KEY=30 THEN
 INVSBL=0:GOTO 830
800 INVSBL=1:GOTO 830
```

Clear the screen, print message, and initialize the word table.

```
830 GRAPHICS 17:SETCOLOR 4,0,4:SETCOLO
R 1,2,8:SETCOLOR 0,12,10:SETCOLOR 2,12
,10
835 POSITION 0,7:PRINT #6;"ONE MOMENT
PLEASE..."
840 FOR I=MN TO MX:FOR J=MN TO MX:T(I,
J)=0:NEXT J:NEXT I
```

Pick a word, its starting position (X,Y), and direction (D). The direction can be 1 thru 8. In a clockwise direction, 1 is west to east, 2 is northwest to southeast, etc. If a word can fit in the table, store it. Otherwise, pick another word.

```
850 FOR I=1 TO NW
860 R1=INT(RND(0)*5)+1:R2=INT(RND(0)*1
0)+1
865 IF NW=5 THEN RESTORE R1+409:GOTO 8
80
870 RESTORE R1+419
880 FOR J=1 TO R2:READ T$:NEXT J
890 W$(12*I-11)=T$
900 IF I=1 THEN 930
910 FOR J=1 TO I-1:IF W$(12*J-11,12*J+
LEN(T$)-12)=T$ THEN POP :GOTO 860
920 NEXT J
930 T$=" ":J=1
940 P=12*I-12+J:IF W$(P,P)="*" THEN 96
0
950 T$(J,J)=W$(P,P):J=J+1:GOTO 940
960 X=INT(RND(0)*(MX-MN+1))+MN:Y=INT(R
ND(0)*(MX-MN+1))+MN
970 D=INT(RND(0)*8)+1:X(I)=X:Y(I)=Y:D(
I)=D:L(I)=LEN(T$):GOSUB 20:GOSUB 40
980 IF E=1 THEN 860
990 C(I)=0:NEXT I
```

Print thank-you message. Then fill the screen with the selected words along with randomly chosen letters.

```
1000 GOSUB 390:POSITION 4,9:PRINT #6;"
THANK YOU."
1010 FOR I=1 TO 400:NEXT I
1020 POSITION 0,7:PRINT #6;"
          ":POSITION 4,9:PRINT #6;"
      "
1030 SETCOLOR 0,0,0
1040 GOSUB 130
1050 IF INVSBL=1 THEN FOR I=1 TO 400:N
EXT I:GOSUB 390:SETCOLOR 0,0,4
```

Print the headings for the bottom 3 lines.

```
1140 POSITION 0,21:PRINT #6;"word";CHR
$(26)
1150 POSITION 0,22:PRINT #6;"remaining
";CHR$(26)
1160 POSITION 0,23:PRINT #6;"value";CH
R$(26):POSITION 10,23:PRINT #6;"score"
;CHR$(26)
```

Initialize and print the number of words remaining (R), value (V), and score (S).

```
1170 R=NW:V=20:S=0:W=S:M=S:X=9:Y=X:POK
E 18,S:POKE 19,S:POKE 20,S
1180 LOCATE X,Y,C:POSITION X,Y:PRINT #
6;CHR$(C+160)
1190 GOSUB 180:GOSUB 190:GOSUB 200
```

If no words are left, go to the end-of-game routine.

```
1200 IF R=0 THEN GOTO 1590
1205 IF W=0 THEN 1210
```

Print the next word that has not been found.

```
1206 IF C(W)=1 THEN V=V*2:GOSUB 190
1210 W=W+1:IF W>NW THEN W=1
1220 P=12*W-11:IF W$(P,P)="*" THEN 121
0
1230 POSITION 6,21:PRINT #6;"
  ":POSITION 6,21
1240 IF W$(P,P)="*" THEN 1255
1250 PRINT #6;CHR$(ASC(W$(P,P))+160);:
P=P+1:GOTO 1240
1255 IF C(W)=1 THEN GOSUB 300
```

Clear register containing the last key pressed.

```
1260 POKE 764,255
1270 IF (C(W)=1 AND V=1) OR (C(W)=0 AN
D V=2) THEN 1310
```

Reduce the point value if one minute has elapsed. Location 20 is incremented by 1 every 1/60 of a second. When it gets to 255, location 19 is incremented by 1.

```
1280 IF 256*PEEK(19)+PEEK(20)<3600 THE
N 1310
1290 GOSUB 390:POKE 19,0:POKE 20,0
1300 IF C(W)=1 THEN V=V-1:GOTO 1305
1302 V=V-2
1305 GOSUB 190
```

If the space bar is pressed (KEY = 33), find the next word. If "G" is pressed (KEY = 61), do the give-up routine. If "C" is pressed (KEY = 18), print a clue and cut the point value in half.

```
1310 KEY=PEEK(764)
1320 IF KEY=33 THEN FOR I=1 TO 30:SOUN
D 0,121,12,10:NEXT I:SOUND 0,0,0,0:GOT
O 1200
1330 IF KEY=61 THEN A=32:GOSUB 395:GOS
UB 230:GOTO 1200
1335 IF KEY=18 THEN POKE 764,255:IF C(
W)<>1 THEN GOSUB 300
```

If the trigger is pressed, check to see if it is a hit or a miss. For a miss, print the message and end the game on the third miss (lines 1370-1420). For a hit, highlight the word and adjust the score (line 1440).

```
1340 IF STRIG(0)<>0 THEN 1460
1350 IF X(W)=X AND Y(W)=Y THEN 1440
1370 M=M+1:POSITION 3,20:PRINT #6;"MIS
S NUMBER ";CHR$(ASC(STR$(M))+128)
1380 GOSUB 380:POSITION 3,20:PRINT #6;
"                 "
1390 IF M>2 THEN A=32:GOTO 1590
1400 POSITION 3,20:PRINT #6;"
    "
1410 IF STRIG(0)=0 THEN 1410
1420 GOTO 1270
1440 GOSUB 370:A=128:GOSUB 230:S=S+V:G
OSUB 200
1450 GOTO 1200
```

Move the cursor in the direction indicated by the joystick.

```
1460 ST=STICK(0):IF ST=15 THEN 1270
1470 IF ST<8 THEN X1=1:Y1=ST-7:GOTO 15
00
1480 IF ST<12 THEN X1=-1:Y1=ST-11:GOTO
 1500
1490 X1=0:Y1=ST-15
1500 IF Y1=-2 THEN Y1=1
1510 IF X+X1<MN OR X+X1>MX OR Y+Y1<MN
OR Y+Y1>MX THEN 1270
1520 POSITION X,Y:PRINT #6;CHR$(C)
1530 X=X+X1:Y=Y+Y1:LOCATE X,Y,C
1540 IF C<91 THEN D=C+160:GOTO 1570
1550 IF C<123 THEN D=C+128:GOTO 1570
1560 D=C+32:GOTO 1570
1570 POSITION X,Y:PRINT #6;CHR$(D)
1580 GOTO 1270
```

Highlight words not found.

```
1590 FOR W=1 TO NW
1600 P=12*W-11:IF W$(P,P)="*" THEN 162
0
1610 GOSUB 395:GOSUB 230
1620 NEXT W
```

Display flashing letters and play-again message. If yes (KEY = 43), return to the level selection routine. If no (KEY = 35), display the final message and end.

```
1625 POSITION 2,20:PRINT #6;"PLAY AGAI
N (Y-N)":POKE 764,255
1630 POSITION X,Y:PRINT #6;CHR$(C):C=0
1640 IF C=2 THEN C=0:L=0:GOTO 1655
1650 C=2:L=8
1655 GOSUB 390:SETCOLOR 0,C,L
1660 FOR I=1 TO 200
1665 KEY=PEEK(764):IF KEY=43 THEN POP
:GOSUB 390:GOTO 710
1670 IF KEY<>35 THEN NEXT I:GOTO 1640
1680 POP :GOSUB 390
1690 GRAPHICS 17:SETCOLOR 4,0,4:SETCOL
OR 0,12,10
1700 POSITION 0,10:PRINT #6;"THANKS FO
R THE GAME."
1710 FOR I=1 TO 400:NEXT I:POKE 764,25
5:GOSUB 390
```

ATARI® Software

A review by Bruce Chapman

Ghost Hunter
from Arcado PLUS. 16K,
cassette, $29.

Crush, Crumble and Chomp
from Epyx. 32K, disk, $32.

Space Chase
from Swifty Software. 16K,
cassette, $15.

Outdoor Games
from Sensational Software. 32K,
cassette, $12.

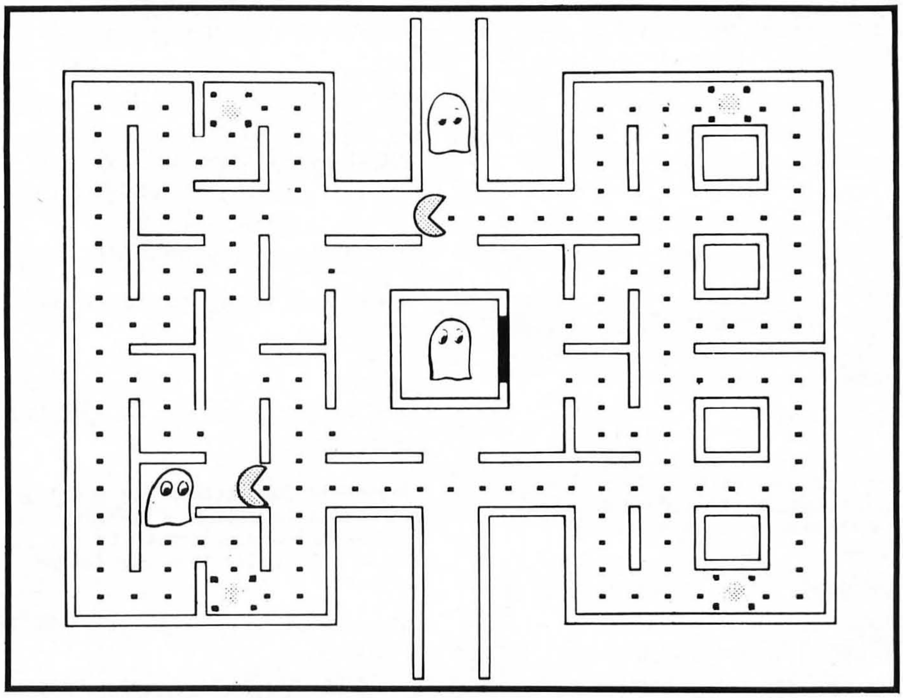
Of these four programs, two are outstanding values — *Ghost Hunter* and *Crush, Crumble and Chomp* (from here on referred to as *CC&C*).

Ghost Hunter, an excellent version of *Pac-Man*, is definitely worth its $29 suggested retail price. It has excellent color graphics, great sounds, incredible action, and many variations including head-to-head playing, player handicapping, and different mazes. The tape version loads in only three minutes with a front screen (which loads in a few seconds) being displayed while the remainder of the program is loaded. The program will auto-start once loaded by scrolling up the first maze from the bottom of the screen while playing a fanfare of some pretty good sounding four-part music. You then can select a maze, choose any options you want, enter your initials, and start play.

The one-player game is at least as fast-paced and difficult as the arcade game, with a few differences. Most important, if you know any "patterns" on *Pac-Man*, you'll have to learn some new ones for this game; the 16 different mazes make it almost impossible to become so good that the game is boring.

Because the joysticks have eight positions rather than the four on the *Pac-Man* machine, it is not as simple as the arcade game. Once in a while you may misguide your "hunter." To facilitate use of the joysticks, I either put a large book on my lap with the joystick on top of it or put the joystick directly on a desk. This also lessens the tension on the hands. During a game, play may be suspended by the touch of any key. If a key is held down long, play will continue more slowly (because of the repeat key feature built into the ATARI® OS).

GHOST HUNTER



Other unique features of *Ghost Hunter* are: the passageway where you can go off the top or bottom instead of the sides; the "fruit" which is not fruit, but pieces of another "hunter" which, when hit, count as extra points worth one-sixth of a new hunter; a "hide" feature which is sometimes helpful; and the sounds/graphics which, although not exactly like those of *Pac-Man*, are excellent.

The two-player head-to-head option is definitely the feature which puts this game far ahead of most others; two people can control hunters roaming the mazes, helping, tricking, or ignoring each other.

Epyx's *CC&C* is also a very good game — especially in disk. The cassette version, unfortunately, may take up to 15 minutes to load (needs 32K RAM,) and then only loads in stages, thereby requiring your attention to the loading. If your patience is good, the game itself is worth the wait.

CC&C redefines the character set which makes the buildings, cars, tanks, bridges, etc., show up on the screen. Although the graphics are excellent, when objects are moved it looks rather awkward. The monster (you can select from six or design one yourself) is a quadruple-width player/missile character, and looks fairly realistic. But then, who's to say what a "realistic" monster is?

The documentation included is excellent and the most extensive I've had for any software package. It starts by telling you to "Let loose the beast within you!" and describes modes of play and objectives very well. While the program is running, it shows a small section of one of four cities which you have previously selected, and draws the other portions of the city as you reach them. This program is a good strategy builder, and the strategy used must be quite different for different monsters, as each has certain features and limitations.

The only major complaint is that the input of moves takes some getting used to. *CC&C* is a great game for those interested in planning and executing a wave of destruction, and definitely worth the $32 if you have a disk drive. If not, patience is recommended.

Swifty Software's *Space Chase*, on the other hand, is not as well-designed. The basic idea behind the game is good. It would be a popular game provided the program itself were of better quality. You control a spaceship with which you attempt to "cloak" all planets (boxy-looking circles) on the screen while avoiding one to three enemy craft after you.

This could be a fast-action arcade game, but somehow the speed and good graphics were left out. The spaceships are simply redefined characters, which jump from one position to the next in a somewhat jerky manner. The only use of player/missile graphics is in the enemy bases, which don't move. The graphics are flashy, but not very smooth or impressive. Joystick control is jerky. One option which allows you to plant mines in the paths of enemies is almosts fruitless, as is the option to have shields. This game may very well hold your interest for awhile but lacks longtime intrigue. The 16K tape costs $15, which isn't a bad price, but I would not highly recommend this program.

Another cassette, I would not recommend *Outdoor Games*. The cassette contains three programs with one variation to one of them. The first program, *Fishing Trip*, is not well executed or designed. It is quite slow, uninteresting with poor graphics (in mode 3), plus it takes too long to load. *Treasure Island I and II* are somewhat of a challenge, but their graphics aren't very good. A map of an island is randomly drawn, with four "guardians" and one blinking treasure. You are to find your way to the treasure by observing around you, searching for useful items (i.e., a compass, some food), and evading the guardians. Unfortunately you can't see your position on the screen, but must deduce your position through examining surrounding terrain.

Here the idea of the program is good, as is the case with *Treasure Island II*, but the graphics, speed, and input are only fair. The last program on the tape, *Forest Fire*, puts you in control of a helicopter (a white dot) so that you may extinguish the fires displayed on the screen. The graphics are not bad (a modified graphics 10 screen is used), but the game is, not very impressive. You might learn from looking at the listing of the program, but aside from this the program is of little value.

Exploring the ATARI® Frontier

ANTIC and the Display List: Part 1

by Alan J. Zett

Note: It is not recommended that beginners attempt to read this article without first having a thorough understanding of BASIC programming. Some books to read that will give you sufficient background include: *ATARI® BASIC Manual, Introduction to Programming* (Schlaum's Outline Series). It is also recommended that the following book should be read: *6502 Assembly Language Programming* (McGraw-Hill).

The Shape of Things to Come

The aim of this series of tutorials is to cover all aspects of ATARI® graphics capabilities in minute detail so those with the inspiration can learn. Advanced graphics will be the central theme, with occasional excursions into the realm of Machine Language. As more of you learn the secrets, everyone will benefit from the improvements in the quality of ATARI® software.

If you have the knowledge, experience, and ability, the ATARI® will produce far superior games in BASIC than the TRS-80® or Apple. However, the ATARI® is a bit slow and the features that make it superior are, at best, well hidden. Making superior games will not be easy for beginners.

Generally speaking, the ATARI® is comparable to the TRS-80® and the Apple. I believe that the Apple has a distinct advantage in writing good games full of color and detail with simple BASIC coding techniques. The number of colors available in Hi-Res when teamed up with shape tables and the ampersand vector make for amazing displays. The TRS-80® is best for games not requiring sharp graphics and color. Big Five Software and Adventure International have had a lot of success with their lines of arcade and adventure games.

If I had to give a rating to all three computers, it would be as follows:

TRS-80®: Excellent for business. Good for games.
Apple: Very good for business. Very good for games.
ATARI®: Good for business. Potentially excellent for games.

If a programmer knows all of the features the ATARI® is capable of and how to use them, the ATARI® is an ideal game computer. But learning these "undocumented features" can be extremely difficult. Even the most concise manual can be confusing at times. Sometimes, the concepts get totally beyond me. That's usually when I decide to call Atari, Inc.

Atari has done more customer support for their computer than any other company in the business. By providing detailed manuals for users and having a toll-free technical hot line, anyone with the inspiration can become a master at the ATARI®.

ANTIC — What is it?

I will begin this month by introducing you to ANTIC: the ATARI®'s graphics nerve center. ANTIC is just one of the four special-purpose microprocessors contained within the ATARI®. A microprocessor is an integrated circuit that has the capability to make simple logical decisions. Almost all of the capabilities of the ATARI® are handled by custom-made integrated circuits that control literally every aspect of its operation. Most of you are already familiar with the 6502 microprocessor. The 6502 is the Central Processing Unit (CPU) of the ATARI® computer. You can think of it as large time-sharing computer. It can do nothing useful by itself, but it can process and channel the flow of information to and from other computer circuits in many useful ways. One of the most common methods of communication to a microprocessor is by using a set of predefined instructions. Each instruction directs the CPU how to react in a given situation. In this way, groups of instructions (called a program) can be combined to perform a complicated (multi-stage) function called a task.

The BASIC language is an example of a large task. The instructions contained in BASIC explain to the CPU what BASIC is and describe how it works. The CPU follows these instructions and tells other circuits what to do and/or listens to what they have to say.

The 6502 microprocessor has a hundred or so instructions it understands. In addition, some of these instructions require extra information to further explain to the CPU what is to be done. The reason for so many instructions is that the CPU is responsible for communicating to all other circuits in the computer. It must be able to talk to any circuit in the system, no matter how complicated. In this sense, the CPU is a universal translator between all of the components in a computer. But only under the direction of a program can it know how to translate.

When the CPU needs to communicate with the outside world, it contacts other circuitry in the computer that has been designed for that purpose. When the CPU is told via the program to get data from the operator, it requests information be input from the keyboard. When it is told to talk to the operator, it sends information in computer form to an area of memory called "screen memory". ANTIC's main responsibility is to translate and encode this CPU data to a form that can be understood by the video processing circuits.

In a less versatile computer, this information could only be displayed in a fixed, unchangeable manner. This is because most computers contain single-purpose circuits for converting screen data into letters and/or graphics on a video screen. In the ATARI®, the circuitry responsible for this type of work is a microprocessor called CTIA.

ANTIC, on the other hand, is a special-purpose processor for displaying visual information. Its duty is to act as a Graphics Processing Unit (GPU). ANTIC sends CTIA display information based on data in screen memory according to a short "program" written in ANTIC's own language called the "Display List" or the "DL." The GPU program must be able to tell ANTIC where screen memory is, and how to display it.

To further clarify the differences between the 6502, ANTIC, and CTIA, use this analogy. A company that designs buildings is going to build a house for a client. First the client talks to the owner of the company and comes to an agreement on what type of house should be built. The owner then tells the architect to design a house based on the client's description. The architect lays out the design of the house and hires some workers to build it. The workers know how to build, but without directions from the architect, what they build will have no relation to what the client wants. In this sense, the computer is the company which builds houses, the 6502 is the owner of the company, the display list is the client who wants a house to be built, ANTIC is the architect who designs houses, and CTIA is the group of workers who build the house according to ANTIC. The house is the video signal which is shown on the video display.

CTIA will be covered in more detail when the subject of Player/Missile graphics is discussed later in the series.

A TV displays a picture by forcing a beam of electrons in straight lines across a screen lined with phosphor. Each horizontal line of phosphor glowing with electrons is known as a "scan line." A picture is formed by tracing many scan lines beneath one another to form a complete image. On the ATARI® video display, there are 192 scan lines. Note that there are many other scan lines above and below the visible video display. These are to make sure that the display information will not be written too high or too low on the picture tube to be seen.

ANTIC works with sets of scan lines called "mode lines." Mode lines can be any number of scan lines between one and 16. Any number of mode lines may be combined, as long as their total comes out to be 192.

ANTIC sends CTIA display information based on data in screen memory.

BASIC already has a set of nine preprogrammed display lists available to the programmer. These are more commonly known as GRAPHICS 0 to 8. GRAPHICS 0 to 2 are text modes; 3 to 8 are color graphics modes. To better understand what a display list does, we will be dissecting the display list that BASIC generates when you type GRAPHICS 0.

ANTIC has only four types of instructions: graphics mode, character mode, blank mode line, and jump instructions.

Graphics mode instructions tell ANTIC to display a line of screen data as color graphics. Character mode instructions display single or multi-color text. Blank mode line instructions tell ANTIC to skip a number of lines on the screen without printing. Jump instructions are used to continue execution of a display list at the specified memory location.

In addition to these modes, there are four special options that may also be specified in any text or graphics mode: Display List Interrupts (DLI), Load Memory Scan (LMS), Vertical Scroll, and Horizontal Scroll. For the purpose of discussing the GRAPHICS 0 display list, the general use of the LMS option will need explanation. All of the options will be covered more extensively in future discussions of display lists, but for now we'll stick to basics.

A Sample Display

BASIC's GRAPHICS 0 consists of 24 screen lines of ANTIC mode 2. Each mode line consists of eight individual scan lines. If we multiply the number of scan lines by the number of screen lines, we come up with a total of 192 scan lines. Figure 1 is the list of all ANTIC text and graphics mode instructions with other relevant information. There are many other forms of these instructions that do not pertain to this month's column and have been left out. You may notice that there are certain ANTIC modes that have no corresponding BASIC GRAPHIC mode. This is because ANTIC has three text modes and two graphics modes which are unused in BASIC. However, this does not mean they cannot be used as we'll see in the future.

First of all, we have to know where to find the display list. This is conveniently provided in memory locations 560 and 561. Every time a GRAPHICS command is executed, the Operating System (OS) cartridge builds a display list in memory and puts its address in these locations. From then on, this is where ANTIC will look to find the display list. To find the actual memory location, use this formula: DL = PEEK(560) + PEEK(561) * 256. The variable DL will then hold the starting location.

Now that we know where to find the display list, lets have a look at it. Figure 2 is a listing of a standard GRAPHICS 0 display list. Keep in mind that what you are looking at is actually a Machine program written in ANTIC Machine Language. Starting from the top of the display list and working down, we first see three bytes of 112. (A byte is roughly one number with a value in the range of 0 to 255.) Each of these bytes is of a special class known as blank mode lines. Figure 3 is a table of blank mode line codes.

This type of ANTIC instruction is responsible for skipping a number of scan lines on the video display. One of its main uses is in bringing the start of the video display down far enough so that the top is clearly visible on the video screen. The blank mode line's other main use is to separate sections of the display without using up screen data. That is, to insert blank lines without having to provide blank screen data. The blank mode instruction 112 causes eight scan lines to be skiped. Three 112 codes will cause the top of the video display to start 24 scan lines down from the top of the video screen. (The video screen starts at the top of the picture tube.) In almost every case, a display list should start with three 112 instructions.

Figure 1: ANTIC Text and Graphics Mode Instructions

ANTIC DL Mode No.	ATARI® BASIC Graphics Mode	No. Scan Lines Per Mode Line	Screen Bytes Per Mode Line	No. of Colors	Type of Mode Line
0		(see Figure 3)			Blank
1	-	-	-	-	Jump
2	0	8	40	2	Text
3	n/a	10	40	2	Text
4	n/a	8	40	4	Text
5	n/a	16	40	4	Text
6	1	8	20	5	Text
7	2	16	20	5	Text
8	3	8	10	4	Graphics
9	4	4	10	2	Graphics
10	5	4	20	4	Graphics
11	6	2	20	2	Graphics
12	n/a	1	20	2	Graphics
13	7	2	40	4	Graphics
14	n/a	1	40	4	Graphics
15	8	1	40	2	Graphics

After this, the fourth byte we find is a 66 instruction. This one is a little tricky. The byte we see is actually the ANTIC mode line 2 instruction (GRAPHICS 0 line) with 64 added to it. Whenever you take a text or graphics mode line value and add 64 to it, you are telling ANTIC that you are using the Load Memory Scan (LMS) option. What the 66 byte is actually telling ANTIC is this: The first line of the display is a GRAPHIC 0 text line and the two bytes which follow it are the memory location of the start of screen memory. ANTIC then uses this address as the start of computer data to be interpreted. The first text or graphics mode instruction should always contain the LMS option.

Bytes 5 and 6 are the screen memory address in Least Significant Byte (LSB) / Most Significant Byte (MSB) order. If either of these terms is unfamiliar to you, I suggest you visit your local computer store or library and pick up a book on the computer numbering system. But for now let it suffice that the MSB is the integer portion of any number in the range of 0 to 65535 which is divided by 256 ((MSB = INT(n/256)). The LSB is the remainder after the MSB * 256 is subtracted from the original number. I hope this explanation is helpful because the concepts of bytes and LSB/MSBs are essential to understanding any computer on its own level (the machine level) i.e., Machine Language.

Bytes 7 through 29 tell ANTIC that there are 23 more lines of mode 2 (GRAPHICS 0) to follow. These mode 2 bytes along with the first mode 2 byte (with the LMS option) combine to form the 24 text lines of GRAPHICS 0.

Byte 30 is a special form of the jump mode. Normally if a mode line 1 instruction is specified, ANTIC does the equivalent of a BASIC GOTO command. This is necessary because, like any other program, the last instruction in the display list must either jump somewhere to continue or end execution. Since the video display is in use almost constantly, the display list should form an endless loop. However, if the display list were to continue executing itself immediately after the last instruction, the first line of the next screen display would appear directly BELOW the last line of the previous display. When ANTIC tries to continue this program, CTIA will get confused and the video screen will begin to flip very fast. This would happen because the TV set or video monitor would be trying to write information out of syncronization with the standard video signal sequence. In other words, video data would be confused with positioning data in the composite video signal coming from CTIA. To make sure that the display starts at the top of the screen in the same place as before, the Wait for Vertical Blank (WVB) option for jump commands is used.

If you add 64 to a jump instruction (mode line 1), ANTIC will halt execution of the display list until the video has started a new picture. This type of instruction is always required at the very end of a display list.

The last two bytes, 31 and 32 are the LSB/MSB of the start of the display list in memory. Well, that certainly was a workout, but from the ATARI®'s point of view, we've only just begun.

The real versatility of ANTIC starts to shine through when we combine different modes in the same display list. Take GRAPHICS 5, for example. There are 40 lines of graphics followed by 4 lines of text.

Figure 2: A GRAPHICS Zero Display List

DL BYTE NO.	BYTE VALUE	MODE TYPE
1	112	Blank
2	112	Blank
3	112	Blank
4	66	Text w/LMS option
5	nn	Least significant byte of screen memory
6	nn	Most significant byte of screen memory
7	2	Text
8	2	Text
9	2	Text
10	2	Text
11	2	Text
12	2	Text
13	2	Text
14	2	Text
15	2	Text
16	2	Text
17	2	Text
18	2	Text
19	2	Text
20	2	Text
21	2	Text
22	2	Text
23	2	Text
24	2	Text
25	2	Text
26	2	Text
27	2	Text
28	2	Text
29	2	Text
30	65	Jump w/WVB option
31	nn	Least significant byte of DL
32	nn	Most significant byte of DL

The Display Mix Twist

Figure 4 is a display list for GRAPHICS 5. As usual, there are three blank mode line instructions at the start for lowering the top of the video display. Following this is a graphics mode 10 byte with the LMS option specified (10 plus 64). The chart in Figure 1 shows that mode 10 is the same as GRAPHICS 5 so we're on the right track. Bytes 5 and 6 are the start of screen memory address that go with the LMS option of byte 4. Bytes 7 to 45 are the other 39 graphics mode lines we would expect to find.

Now look at byte 46. If you remember, a 66 in the GRAPHICS 0 display list represented a mode 2 line with the LMS option. Here is where ANTIC splits the video display.

According to Figure 1, each line

Figure 3: ANTIC Blank Mode Instructions

ANTIC DL Mode Byte	No. of Blank Scan Lines
0	1
16	2
32	3
48	4
64	5
80	6
96	7
112	8

EXPLANATION OF ATARI® LINE LISTINGS

SoftSide uses the following conventions in representing unprintable characters in ATARI® line listings, unless otherwise noted:

Characters (including blank spaces) which are underlined should be typed in inverse video.

When graphics or control characters are to be included in a string (between quotation marks), that fact will be noted in a nearby REMark. In such cases, graphics characters are represented by the corresponding lower-case letter, and control characters are represented by the corresponding unshifted key symbol. For example: The lower-case letter s represents a graphics cross, entered by holding down the CTRL key and then pressing the S key. The symbol = represents a control-down-arrow, entered by first pressing and releasing the ESC key, then holding down the CTRL key and pressing the = key. (See Appendix F, and the back cover, of the *ATARI® BASIC Reference Manual.*)

The one regular exception to the above is that a clear-screen character (ESC CTRL-   ) is represented in listings by a right-hand brace, which looks like this: #

A shifted = is represented in the listings by a vertical line with a small gap in it.

Figure 4: A GRAPHICS 5 Display List

DL BYTE NO.	BYTE VALUE	MODE TYPE
1	112	Blank
2	112	Blank
3	112	Blank
4	74	Graphics w/LMS option
5	nn	Least significant byte of screen memory
6	nn	Most significant byte of screen memory
7	10	Graphics
8	10	Graphics
9	10	Graphics
.	.	.
.	.	.
.	.	.
.	.	.
43	10	Graphics
44	10	Graphics
45	10	Graphics
46	66	Text w/LMS option.
47	nn	Least significant byte of screen memory
48	nn	Most significant byte of screen memory
49	2	Text
50	2	Text
51	2	Text
52	65	Jump w/WVB option
53	nn	Least significant byte of DL
54	nn	Most significant byte of DL

of GRAPHICS 5 takes up 20 bytes of screen memory. When ANTIC comes to byte 46, it reloads the screen memory pointer to start at the address contained in bytes 47 and 48. As long as the data here are text data, ANTIC will set up the next video mode line to be in GRAPHICS 0. The next 40 bytes of data at the location specified in bytes 47 and 48 will be displayed as one line of text.

Bytes 49 to 51 tell ANTIC that the next three lines are also GRAPHICS 0. No other LMS instruction was used since the last LMS option was specified, therefore ANTIC will continue to look at the data following the first GRAPHICS 0 line.

Bytes 52 to 54 perform the same function as bytes 30 to 32 in Figure 2.

Now, you might be asking yourself why there are only four GRAPHICS 0 lines after the GRAPHICS 5 lines. If you recall, earlier I mentioned that each video display must be a maximum of 192 scan lines long. If you look at Figure 1, you will see that each GRAPHICS 5 mode line requires four scan lines while each GRAPHICS 0 requires eight. If you multiply 4 text lines times 8 scan lines, then add 40 graphics lines times 4 scan lines, the answer comes out to 192 scan lines and ANTIC is satisfied.

The Catch!

Well, as you might have figured, there is one small problem to be overcome when writing display lists. ANTIC can only look at consecutive screen memory as long as it is 4K bytes long or less. (Beginners skip this next sentence.) This is because ANTIC's program counter is only 12 bits wide, which means 4K is its incrementing limit. (Welcome back beginners.) However, the memory requirement for GRAPHICS 8 is 7.5K. So (the intrepid beginner asks), how then does ANTIC display 7.5K of data when 4K is its limit?

I'm glad you asked that! It shows you're thinking. Refer now to Figure 5 where you will find a GRAPHICS 8 display list. The bytes from 1 to 99 should be familiar to you by now. But look at bytes 100 through 102. Haven't we seen them somewhere before?

Look at bytes 3 to 5. They both start with the same mode byte. What bytes 100 to 102 do is tell ANTIC that in order to finish with the display, it must continue looking for screen data at the address contained in bytes 101 and 102. By literally hopping over the 4K boundary, ANTIC can display data as long as desired. Byte 100 is simply a GRAPHICS 8 mode line byte with the LMS option specified. All other bytes from 103 to 176 have been explained before.

Well, that covers most of the basics. In my next installment, I'll be covering the text and graphics mode line options in more detail as well as making a few custom display lists. In the meantime, experiment with the program in Figure 6. It will allow you to look at the display list of any BASIC GRAPHICS mode from 0 to 56. If you work at it, by the next column you should be able to recognize any NORMAL display list.

In the interest of aiding all ATARI® users, reader input will be passed on whenever possible. Any comments, ideas, corrections (knock on wood), or information that will help me to convey a correct and simple explanation of the ATARI® graphics features will be welcome.

Figure 5: A GRAPHICS 8 Display List

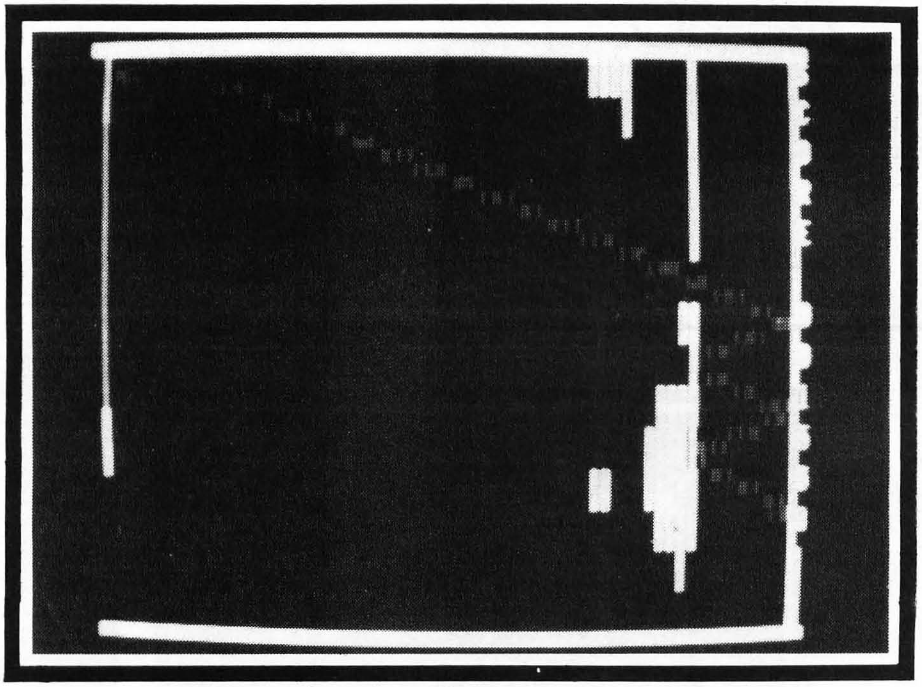
DL BYTE NO.	BYTE VALUE	MODE TYPE
1	112	Blank
2	112	Blank
3	112	Blank
4	79	Graphics w/LMS option
5	nn	Least significant byte of screen memory
6	nn	Most significant byte of screen memory
7	15	Graphics
8	15	Graphics
9	15	Graphics
.	.	.
.	.	.
.	.	.
97	15	Graphics
98	15	Graphics
99	15	Graphics
100	79	Graphics w/LMS option
101	nn	Least significant byte of screen memory
102	nn	Most significant byte of screen memory
103	15	Graphics
104	15	Graphics
105	15	Graphics
.	.	.
.	.	.
.	.	.
165	15	Graphics
166	15	Graphics
167	15	Graphics
168	66	Text w/LMS option.
169	nn	Least significant byte of screen memory
170	nn	Most significant byte of screen memory
171	2	Text
172	2	Text
173	2	Text
174	65	Jump w/WVB option
175	nn	Least significant byte of DL
176	nn	Most significant byte of DL

Figure 6: A Display List Dumping Program

```
10 GRAPHICS 0: CLR: DIM A(202)
20 POSITION 2,11: TRAP 10
30 ? “WHAT GRAPHICS MODE ”;
40 INPUT A
50 GRAPHICS A: FLAG = 0
60 DL = PEEK(560) + PEEK(561) * 256
70 FOR X = 0 TO 202
80 P = PEEK(X+DL)
90 IF P = 65 AND FLAG = 0 THEN
   FLAG = X + 2
100 A(X) = P: NEXT X: GRAPHICS 0
110 ? ,“USE ‘CTRL 1’ TO FREEZE.”
120 ?: ?: FOR X = 0 TO FLAG
130 ? ,“DL BYTE ”;X+1;“ = ”;A(X)
140 NEXT X
150 ?: ?: ?: GOTO 30
```

SoftSide TRS-80® DV

BREAKTHRU



by Larry Meister

***Breakthru* is a Machine Language arcade game for the TRS-80® Model I or III. Optional sound is available for an external amplifier through the AUX cable of the cassette port.**

The object of this game is to achieve as high a score as possible by hitting as many targets as possible. You maneuver the paddle toward the ball so that it intercepts the ball and deflects it back to the target area.

Each target set consists of 140 targets, arranged as ten columns of fourteen blocks each. A block's value is determined by its column. These values are:

Column	Points
First	1
Second	2
Third	4
Fourth	7
Fifth	11
Sixth	16
Seventh	22
Eighth	29
Ninth	37
Tenth	46

This gives a possible total of 2450 points per rack.

Title Display

After the program loads, it will display the the title "BREAKTHRU 4.1 with sound". At this point, you can type the number of balls you want per game, and run with all the standard options set to their default values. Defaults are:

1 - Paddle will shrink if ball hits back wall.
2 - Rack is reset when ball is outside target area.
3 - Speed of ball is fast.
4 - Target is solid block.

Setting Game Options

When the program asks "# Balls Per Game (1-8)?", you can press any number from 1 to 8 to signify the number of balls per game you want, and begin play. The following options are available from the title display:

"H" - Go to Help display.
"P" - Go to Paddle display.
"R" - Go to Reset display.
"S" - Go to Speed display.
"T" - Go to Target display.

Help Display

This command lists all available options. Press any letter shown in parentheses to go to the corresponding display, or press any number from 1 to 8 to specify the number of balls per game and begin play.

Paddle Display

This controls the action taken when the ball hits the back wall. Press "1" or ENTER and, for the duration of that ball, the paddle will shrink if the ball hits the back wall. Press "2" and the paddle will maintain its normal size throughout the game.

Reset Display

This controls the action taken when the last target has been hit. Press "1" or ENTER and a new rack of targets will appear after the ball clears the target zone. Press "2" and the rack is reset immediately after the last target has been hit.

Speed Display

The speed of the ball is chosen here. Press "1" or ENTER for fast speed, "2" for slow speed (approximately 25% slower), and "3" for variable speed.

Target Display

This is used to pick your target. Press ENTER for the standard full brick, or type 129 through 191 for a custom target rack. The number chosen corresponds to the standard TRS-80® block graphic codes. (A big favorite here at ***SoftSide*** is a target character of 149. Try a few games and you'll see why.)

Play Display

To serve the ball - press "S".
To move the paddle up - hold down ">"
To move the paddle down - hold down "<"

When the program asks "Another Game?", you may:

Press "Y"- Start another game with the same options.
Press "N"- Return to the Title Screen.

Cassette Owners!

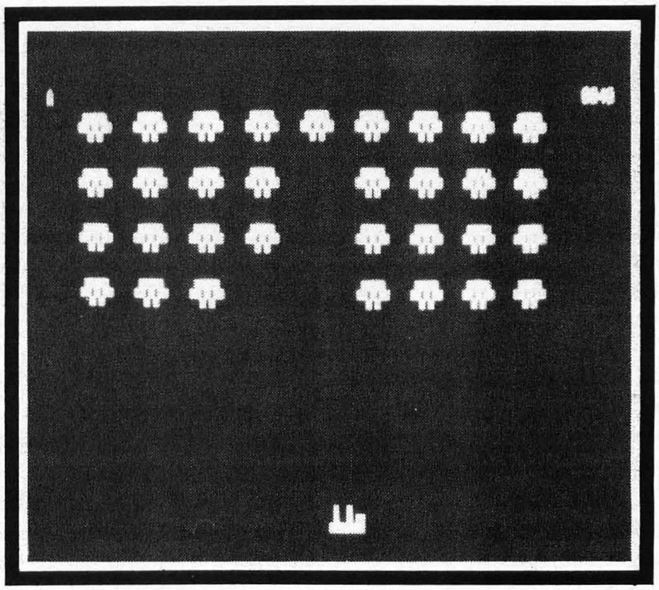
The author wishes to inform everyone that a cassette version of *Breakthru* is available directly from him for $14.00. The tape will run on either Model I or Model III (on the low speed cassette load) and requires 16K. This is handy for readers of ***SoftSide*** who do not own a disk system. The address for ordering is:

Interactive Computer Development
38 Linden Street
Arlington, MA 02174
(617) 646-8063

K-Byter

Mini-Invaders

A TRS-80® K-Byter by V. A. DeGiorgio, Orlando, FL



This is a fast, real-time graphics game consisting of a Machine Language program which is POKEd into memory by a 1K BASIC loader routine. As you type in the program, pay very close attention to the DATA statements since they are in an unusual format. Be sure to SAVE the program before RUNning it, since there is no provision for returning to BASIC.

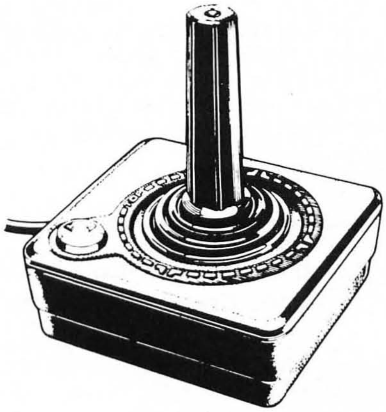
When you RUN the program there will be a delay of about fifteen seconds while the Machine Language routine is initialized. (A message such as "Initializing..." could be added, but then the program would exceed 1024 bytes.) Once the game begins, the upper left-hand corner of the screen will display a graphic representation of the number of bases remaining (including the current one): solid blocks on a Model I, or graphic symbols on a Model III. Your score, one point for each invader destroyed, is displayed at the upper right.

Use the arrow keys to move your base left and right, and the spacebar to fire. Only one missile may be on the screen at a time, so aim your shots. When all four of your bases are destroyed, the game is over; you may restart by pressing the BREAK key.

```
0 CLS:CLEAR300:DEFSTRS:ONERRORGOTO6:A=3E4:IFPEEK(16396)=201THENP
OKE16526,48:POKE16527,117ELSEDEFUSR=A
1 READS:FORI=1TOLEN(S)STEP2:POKEA,(ASC(MID$(S,I,1))-48)*16+ASC(M
ID$(S,I+1,1))-48:A=A+1:NEXT:GOTO1
3 DATA"3>300604210075702З<=;776210575:?0606<=;7763<0624<=;7763>2
0322?7506042;7<;520?;10?9060:<53>802100507711015001??03>=;02:077
5114050=?300321400011<0??192207753>8?77212?753:4038477><;6828043
=280135<;702805?>3=280134<;78280?1108751:;720083>53121;7><6<012

4 DATA"2146503:0675856?110;750>040609<51:;7>52825=511<<76<=;<76=
1<;412819:?3208751206042104757>3<?>3:28037718063>30772;10?1>1060
62310?=13<110<;064:2310?=0=20<12:09757<;7281<110054=?380321<1??1
13?00193:0575<;4728022323220975З>;<77210175113<500>04>=;021;?53

5 DATA"3:2?75472310?=11=076<=;<76<521??5311??3?010004>=;8>;23110
0751:473>??<=;776<1<;41<12811>;35200:3:4038?>0420?9<33075<34;750
5<25;75210;750124003>01>=;1<23<752106757>2;<;46280511010618031l?
??:;:2005347;>=445?237>83773:0:75;7<25975>=5?472141523>802;;>28

6 X=USR(0):DATA"?<10?:114050=?380711400019220975<359757723l0?<<9
0104047>?>8020020=1:7723131O?4<98>9;9;84;:;0;5;<
```

Joystick Modification for the TRS-80®

By Tigre Wenrich



Save money by attaching an ATARI® joystick to your TRS-80® yourself.
Here's how:

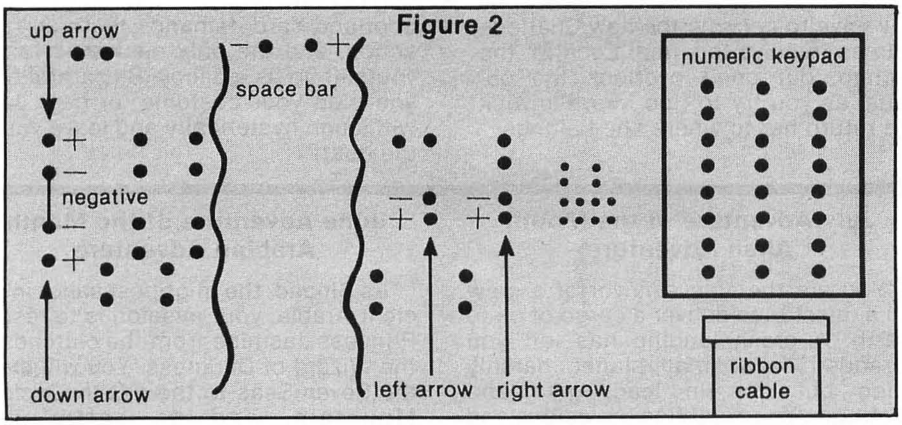
One advantage that the Apple and ATARI® computers have over the TRS-80® is their game playing capabilities. Part of this is due to the lack of a joystick for the TRS-80® Recently a few companies have come out with joysticks using an interface that connects to the computer's expansion port, but these are expensive (about $40). Not only that, but they make use of the INP instruction so programs must be modified or custom-made to make use of the joystick.

I have found that it is not too difficult to attach an ATARI® joystick (the same kind used by other companies with their interfaces) to the TRS-80® keyboard. The joystick is sold by Atari, Inc. as a replacement part for their home video games and costs about $10. I attached it to take the place of the up, down, right, and left arrow keys and the spacebar. This way the joystick can be used with any game that would have used these keys (all but one of the excellent games by Big Five Software work excellently!). BASIC programs can be tailored to use these keys as well. (*Titan* from the December issue of ***SoftSide*** is a good example. Just change statements which use the ENTER key to use the spacebar). The keyboard's operation will not be effected by the joystick in any way. I have found that programs using the INKEY$ statement work slightly better than those using PEEK.

Figure 1

wire color	terminal post
brown	right arrow, positive
green	left arrow, positive
orange	spacebar, positive
white	up arrow, positive
blue	down arrow, positive
black	any negative post (I used down arrow for convenience)

Figure 2



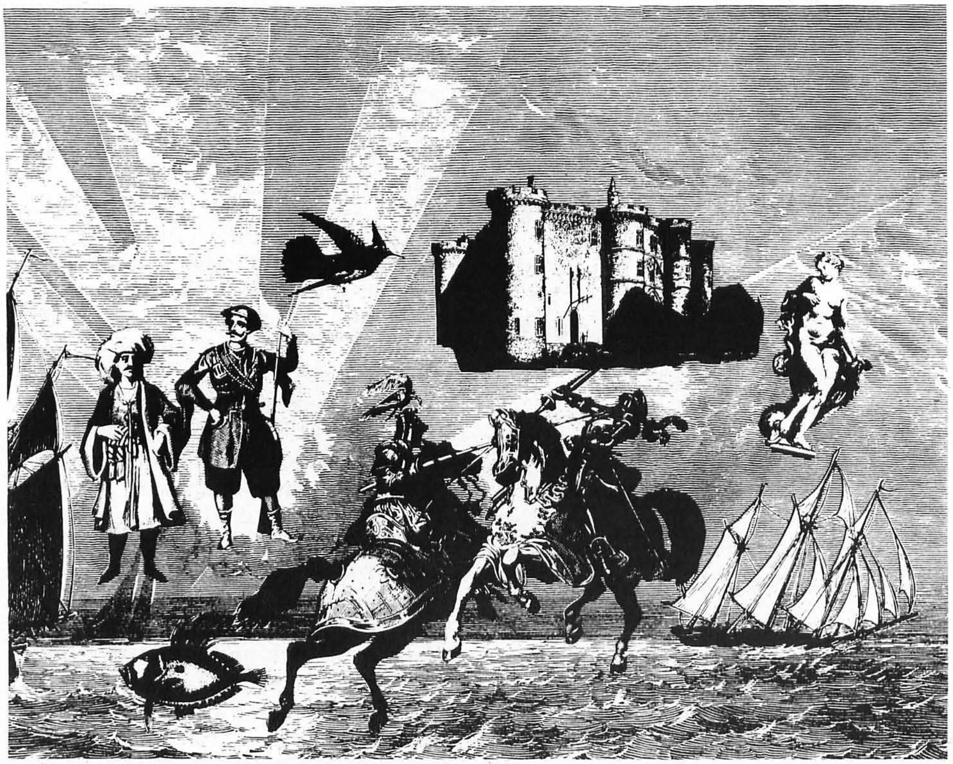
A generalized layout of the underside of the Model III keyboard (The Model I should look the same.)

The Juicy Part

You must start by taking apart your computer case (just the CPU case if you have a Model I). Flip the keyboard over so you can access the soldered terminal points. Clip the plug from the joystick's cord and feed the wire through a space used by one of the expansion cards (it may be necessary to add some extra wire to allow a longer cord for playing). Now solder each of the six wires to the keyboard as shown in Figure 1.

I have done this to my TRS-80® Model III and love the results. I would think that the Model I modification would be no different.

The Adventure is



April Adventure of the Month
Witches' Brew Adventure

You find yourself in an enchanted forest. You must find your way to the castle and rescue the Princess who is chained inside its dungeon. A tightly-woven blend of fantasy, horror, and science fiction, this complex adventure will challenge your wits and ingenuity.

March Adventure of the Month
James Brand Adventure

The President's life is in danger. As James Brand, you must save his life and destroy the evil Dr. Death. Your life is constantly on the line; each move you make could be your last. "Your assignment, Mr. Brand...."

February Adventure of the Month
Klondike Adventure

Snow, ice, and bitter cold surround you. Your search for fame and fortune in the northern country will lead you through many perils, but you may also see some familiar faces along the way.

January Adventure of the Month
Windsloe Mansion Adventure

A famous prisoner lies in the dungeon of an old mansion. An underground passage connects with the Blair house, whose owners will help you to rescue the prisoner. Can you overcome the human and supernatural creatures who inhabit Windsloe Mansion?

December Adventure of the Month
Black Hole Adventure

The crew of an interstellar craft discovers the long-lost Deep-Space Probe One, the Cygnus, at the edge of the vortex surrounding an immense black hole. See if you can foil the plans of Dr. Hans Reinhardt.

November Adventure of the Month
Around the World in Eighty Days Adventure

Try to repeat the feat of the classic novel, complete with a balloon and other exciting features of the original adventure. Are you ready to take the challenge? Bon voyage!

October Adventure of the Month
Crime Adventure

Test your skills as a detective by sifting through hundreds of clues. You may have to become the new Sherlock Holmes to solve this one! Look for the strange, but don't overlook the obvious, as you try to find Mrs. Fenwick and return her to where she belongs.

September Adventure of the Month
Jack The Ripper Adventure

Jack the Ripper is running rampant in London and you must stop him! Scotland Yard demands that you take action, and the only answer is to set yourself up as a decoy. Be careful how you plan your costume, or dear Jack will laugh hysterically and leave you in the dust!

August Adventure of the Month
Treasure Island Adventure

You are a hardy adventurer in search of fame, fortune, and whatever else you can get. You find yourself on an island where there is rumor of pirate's treasure. But watch out for the evil magician and the underground torture chamber! You may end up in a spot where all roads coming into it are paved with good intentions. . .

July Adventure of the Month
Alien Adventure

You are the sole survivor of a crew on a mission to deliver a cargo of oil to Earth. A crash landing has left you stranded on a small planet, harshly alien but rich in lead, gold and platinum. You must find provisions and a means of leaving the planet. But beware of the THING that massacred your crew!

June Adventure of the Month
Arabian Adventure

As Sinbad, the mightiest sailor in ancient Arabia, your mission is to rescue Princess Jasmine from the clutches of the Wizard of Darkness. You will cross the Seven Seas to the deadly Cyclops Mountain, and do battle with skeletons, a one-eyed beast, a hairy tarantula and more monsters who try to thwart your noble pursuit.

Waiting for You...

Subscribe to Adventure of the Month

How would you like to go back in time to 19th century London to match wits with Jack the Ripper? Out into space to brave the swirling vortex of a black hole? Into the depths of the ocean, or on a quest to rescue a beautiful princess from the clutches of evil monsters?

You never know where **SoftSide Magazine's Adventure of the Month** might take you. But you can be sure that each month you will experience new delights and new challenges as you receive an original adventure on tape or disk, ready to load into your computer.

The cost? A six-month membership is just $29 for the tape ($4.83 per adventure) or $49 for the disk ($8.16 per adventure). If you're not sure that you can take six full months of excitement, you can order a single tape for $7 or a disk for $10. Or, if you're especially adventuresome, we're offering disks, packed with three great adventures, for only $26 per disk.

Please use the coupon below (or the bind-in card in this issue) to order.

MAY ADVENTURE OF THE MONTH
TITANIC

You are the Captain of the Titanic on her maiden voyage. Suddenly a large white object comes into view through the window. Can you avoid the historic collision? If not, can you save the lives of your passengers and crew?

Adventure of the Month

6 South Street, Milford NH 03055

Yes, I'm ready to start! Send me Adventures —

■ **Six month subscription:**
- □ Cassette — $29
- □ Disk — $49

■ **Individual adventures (please specify)**

____

____

____

- □ Cassette — $7 each
- □ Disk — $10 each

■ **Three adventures on one super disk ($26 each):**
- □ Arabian, Alien, & Treasure Island Adventures
- □ Jack the Ripper, Crime, & Around the World Adventures
- □ Black Hole, Windsloe Mansion, & Klondike Adventures

Please specify which computer:
- □ Apple (req. 24K for tape, 32K for disk)
- □ ATARI® (req. 32K for tape, 40K for disk)
- □ TRS-80® (req. 16K for tape, 32K for disk)

Name ____________________________

Address ____________________________

City/State ____________________ Zip ________

□ Payment enclosed

□ MasterCard □ VISA Name of Cardholder ______________

MC# and Interbank#/VISA# ____________________

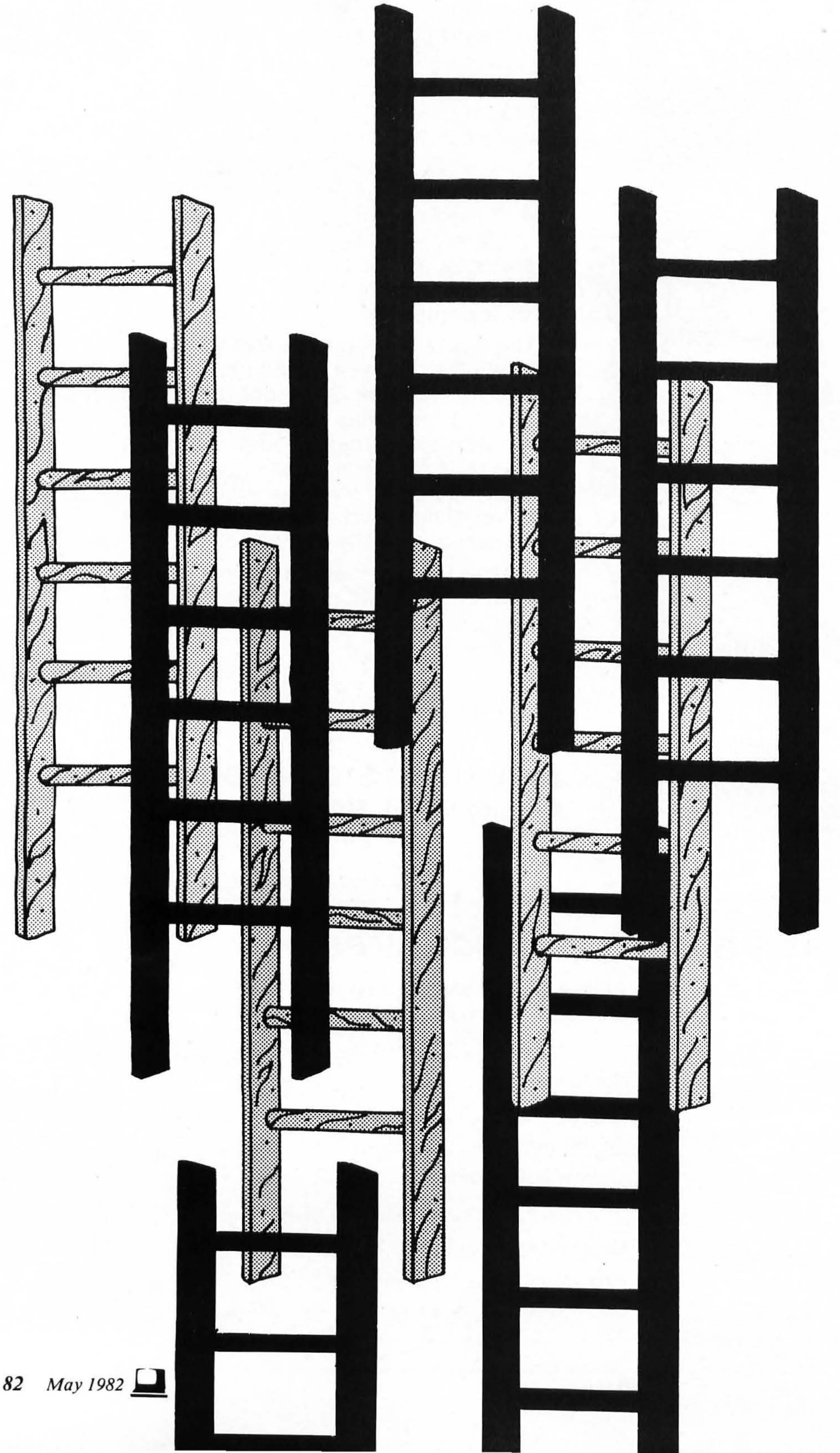
Exp. Date ________ Signature ____________________

Prices subject to change without notice. Apple, ATARI* and TRS-80* are registered trademarks of The Apple Computer Company, Warner Communications and The Tandy Corporation respectively.

LADDERS

by Rik Pierce

Ladders is an original board game for a TRS-80® Model I or III with 16K RAM minimum. DV subscribers will receive an enhanced version with packed graphics and additional sound.



This is a highly challenging game for two, three, or four players. The object is to be the first player to "capture" any three of the eleven ladders displayed on the screen.

Play begins with each player first rolling two dice; whoever gets the highest roll will start. After the first player has been chosen, he begins his turn by rolling four dice. He must group the rolled dice into two pairs, the sum of each pair denoting a particular ladder on the board. The computer places a marker on the lowest rung of the ladders pointed to by the dice. The player may then roll again, pairing the dice as before, either to advance an existing marker one more rung up a ladder or to place another marker on a different ladder. The player may place up to three new markers on the board during his turn.

After each roll, the player is offered the choice to stop his turn or to continue rolling. If the player continues rolling, and comes up with a combination of dice which will not allow him either to place a new marker on the board or to advance an existing one, he loses his turn, and any progression up the ladder is erased.

To "capture" a ladder, the player must reach the top of the ladder and cover the value at the top with his marker. When a player quits his turn, all markers are updated to show their new positions, and any markers at the top will cause the ladder to be filled in, thus wiping out anyone else on the ladder at that time.

To adjust for the fact that some numbers come up more often than others, each ladder is a different height, corresponding to the chance of that number being rolled. This means that ladder seven is the longest, and ladders two and twelve are the shortest.

Careful planning of which ladders to climb and how long to play your turn will decide the outcome of _Ladders_. Good luck!

TRS-80®

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$    TRS-80 Mod I/III BASIC    $
$          "LADDERS"           $
$      Author: Rik Pierce      $
$      (c) 1982, SoftSide      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Reserve high memory according to amount available for Machine Language subroutines.

```
1 CLEAR:POKE16561,111:POKE16562,127-64*(PEEK(16562)>127)
```

Determine if program is running on a Model I or III.

```
2 CLEAR1000:TRS=1:IFPEEK(293)=73THENTRS=3
```

Test for lower-case.

```
3 POKE15360,97:IFTRS=3ORPEEK(15360)<>97THEN10
```

POKE in lower-case driver.

```
4 READA$:IFA$<>"HHH"THEN4ELSEE=PEEK(16562)*256+255
5 E=E+65536*(E>32767):FORI=E-40TOE:READB:POKEI,B:NEXT
7 POKE16414,215:POKE16415,PEEK(16562):RESTORE
```

POKE in sound routine.

```
10 CLS
20 PRINT@386,CHR$(23)"Plug in aux cable for sound"
25 READA$:IFA$<>"MMM"THEN25ELSEE=PEEK(16562)*256+255
30 E=E+65536*(E>32767):FORI=E-131TOE-68:READB:POKEI,B:NEXT
```

Adjust sound routine to look at S$ for notes to play.

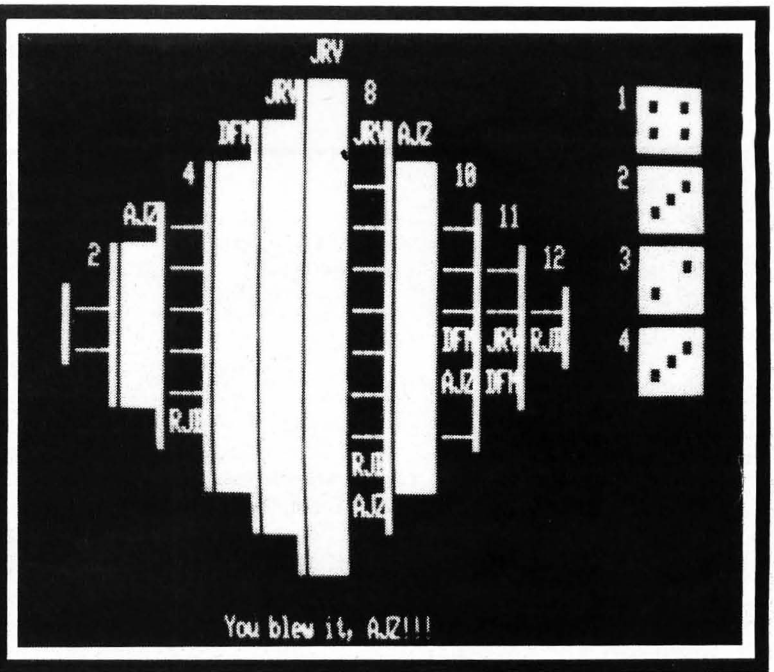
```
40 S$="":V=VARPTR(S$)
50 Q3=INT(V/256):Q4=V-Q3*256:Q0=E-131
70 Q3=Q3-256*(Q3<0):POKEQ0+1,Q4:POKEQ0+3,Q3
```

Set up USR call for tape or disk.

```
80 IFPEEK(16396)=201THENAZ=Q0-65536*(Q0<0):POKE16527,INT(AZ/256)
:POKE16526,AZ-PEEK(16527)*256:GOTO200
90 DEFUSR=Q0:CMD"T":POKE14308,0:RESTORE
```

Define graphic strings and variables, dimension arrays.

```
200 DEFINTA-D,F-Z:DEFSTRC
220 CS=CHR$(31):Q$=CHR$(34)
300 PD=121:PO=896:PB=30:PL=282:PW=PL+388
340 N1=8:N2=7:Z=2:Z1=43
360 DIMR(13),R1(13),L(4,13,13),P(13,14),IV(13),V(13)
400 D$=CHR$(26)+STRING$(6,24):D$(1)=STRING$(2,191)+CHR$(159)+CHR
$(175)+STRING$(2,191)+D$+STRING$(6,143)
420 D$(2)=STRING$(4,191)+CHR$(179)+CHR$(191)+D$+CHR$(143)+CHR$(1
40)+STRING$(4,143)
440 D$(3)=STRING$(2,191)+CHR$(159)+CHR$(175)+CHR$(179)+CHR$(191)
+D$+CHR$(143)+CHR$(140)+STRING$(4,143)
460 D$(4)=CHR$(191)+CHR$(179)+STRING$(2,191)+CHR$(179)+CHR$(191)
```



You blew it, AJZ!!!

```
+D$+CHR$(143)+CHR$(140)+STRING$(2,143)+CHR$(140)+CHR$(143)
480 D$(5)=CHR$(191)+CHR$(179)+CHR$(159)+CHR$(175)+CHR$(179)+CHR$
(191)+D$+CHR$(143)+CHR$(140)+STRING$(2,143)+CHR$(140)+CHR$(143)
500 D$(6)=CHR$(191)+CHR$(179)+CHR$(183)+CHR$(187)+CHR$(179)+CHR$
(191)+D$+CHR$(143)+CHR$(140)+CHR$(141)+CHR$(142)+CHR$(140)+CHR$(
143)
520 GOSUB6140
540 CC=STRING$(3,95):CB=STRING$(3,191):T$(5)=CHR$(140)+CHR$(191)
+CHR$(140)
560 B$=CHR$(149)+STRING$(3,95)
```

PRINT introduction.

```
570 PRINT@0,CS
580 PRINT@390,"Get ready for the game of..."
600 S$=PS$:B=USR(0):PRINT@0,CS
620 FORX=PLTOPL+384STEP64
640 PRINT@X,B$;CHR$(149);
660 NEXT
680 FORX=7TO1STEP-1
700 FORY=7TOXSTEP-1
720 READA$:PRINT@PW+(7-Y)*64-(7-X)*64,A$;
740 NEXTY:RESTORE
760 S$=" "+CHR$(74+X+Y):B=USR(0)
780 NEXTX
800 PRINT@PW+116,"by Rik Pierce"
820 DATA L,A,D,D,E,R,S
840 S$=S1$
```

Offer instructions.

```
860 FORTIM=1TO750:NEXT:PRINT@PO,CS"Need instructions?":B=USR(0):
GOSUB3440:IFA$="Y"THEN F5=5:GOSUB4920:GOTO4020
880 '
```

Enter number of players and their names.

```
900 S$=S2$:PRINT@PO,CS"How many players?":B=USR(0)
920 GOSUB3440:TP=VAL(A$):IFTP<2ORTP>4THEN920
940 FORX=1TOTP
960 IFS$=S1$THENS$=S2$ELSES$=S1$
980 PRINT@PO,CS;"Name of player"X;:B=USR(0):INPUT N$(X)
1000 NEXT
```

```
1020 FORX=1TOTP
1040 T$(X)=LEFT$(N$(X),3)
1060 IFLEN(T$(X))=1THENT$(X)=CHR$(95)+T$(X)+CHR$(95)
1080 IFLEN(T$(X))=2THENT$(X)=T$(X)+CHR$(95)
1100 NEXTX
1120 IFF5=0THENGOSUB4920
```

Roll dice to see who goes first.

```
1160 FORB=1TOTP
1180 PRINT@PO,CS;N$(B)", your roll to see who goes first.  ";
1200 FORY=1TO16:FORX=PD+128TOPD+256STEP128:PRINT@X,D$(RND(6));:N
EXTX:NEXTY
1220 FORX=1TO2:DT(X)=RND(6):PRINT@PD+128*X,D$(DT(X));:NEXT
1240 TT(B)=DT(1)+DT(2)
1260 PRINT@PO,CS,N$(B)", you got"TT(B);
1280 IFTT(B)>HSTHENHS=TT(B):ST=B
1300 FORTIM=1TO1500:NEXT
1320 NEXTB
1340 NP=ST-1
1360 '
```

Start main game loop.

```
1380 NP=NP+1:IFNP>TPTHENNP=1
1400 PRINT@PO,CS,N$(NP)"'S TURN. . .";
1420 S$="(J(7(+<$!+($":K=USR(0)
1440 M=0:F1=0:FORX=1TO3:M(X)=0:NEXT X
1460 FORX=2TO12
1480 IFV(X)=1THENV(X)=0
1500 IFV(X)>1THEN1580
1520 FORY=R1(X)TO1STEP-1
1540 IFL(NP,X,Y)=NPTHENPRINT@P(X,Y),T$(NP);
1560 NEXTY
1580 NEXTX
1600 '
```

Roll four dice.

```
1620 F1=0
1640 FORY=1TO16:FORX=PDTOPD+384STEP128:PRINT@X,D$(RND(6));:NEXTX
:NEXTY
1660 FORX=0TO3:DT(X+1)=RND(6):PRINT@PD-3+128*X,X+1;D$(DT(X+1));:
NEXT
```

Check if legal.

```
1700 DS(1)=DT(1)+DT(2):DS(2)=DT(1)+DT(3):DS(3)=DT(1)+DT(4)
1720 DS(4)=DT(2)+DT(3):DS(5)=DT(2)+DT(4):DS(6)=DT(3)+DT(4)
1740 FM=0
```

Find possible ladders for dice rolled.

```
1760 FORX=1TO6
1780 IFV(DS(X))<2THENFM=1
1800 NEXTX
1820 IFFM=0THEN3460
```

Accept a move.

```
1860 PRINT@PO,CS;"
"N$(NP)", choose a pair of dice -> ";
1880 '
1900 I$=INKEY$:IFI$=""THEN1900ELSEV=VAL(I$):IFV<1ORV>4THEN1900
1920 PRINTCHR$(27)D$(DT(V));" AND ";CHR$(27);
1960 I$=INKEY$:IFI$=""THEN1960ELSEV1=VAL(I$):IFV1<1ORV1>4ORV1=VT
HEN1960
1980 PRINTD$(DT(V1));
2020 S(1)=DT(V1)+DT(V)
2040 IFV(S(1))>1THENGOSUB3020:GOTO1860
```

Gets the value of the second pair of dice.

```
2080 V2=V+V1:IFV2=3THENS(2)=DT(3)+DT(4)ELSEIFV2=4THENS(2)=DT(2)+
DT(4)ELSEIFV2=6THENS(2)=DT(1)+DT(3)ELSEIFV2=7THENS(2)=DT(1)+DT(2
)
2100 IFV2=5THENIFV1=1ORV1=4THENS(2)=DT(2)+DT(3)ELSES(2)=DT(1)+DT
(4)
2120 FL=0
2140 FORZ=1TO2
2160 IFV(S(Z))>1THEN2420
```

PRINT players marker on ladders.

```
2200 FORU=R1(S(Z))TO1STEP-1
2220 IFV(S(Z))>1THEN2320
2240 IFFL=ZTHEN2320
2260 IFL(NP,S(Z),U)>0THENPRINT@P(S(Z),U+1),T$(5);:FL=FL+1:L(NP,S
(Z),U+1)=5:GOSUB3260ELSE2320
2280 IFR1(S(Z))=U+1THENV(S(Z))=2
2300 IFL(NP,S(Z),U)=5THENL(NP,S(Z),U)=0ELSEM=M+1:M(M)=S(Z):IFV(S
(Z))<2THENV(S(Z))=1
2320 NEXTU
2340 IFFL<>ZTHENPRINT@P(S(Z),1),T$(5);:L(NP,S(Z),1)=5:M=M+1:M(M)
=S(Z):FL=FL+1:V(S(Z))=1:NZ=P(S(Z),1)/64:S$="("+CHR$(NZ*3):K=USR(
0)
2360 IFM<3THEN2420
2380 FORX=2TO12:IFV(X)=0THENV(X)=2
2400 NEXTX
2420 NEXTZ
2440 IFFL=0THENGOSUB3020:GOTO1860
```

Offer chance to quit while ahead.

```
2460 PRINT@PO,CS,N$(NP)", do you want to continue?":GOSUB3440
2480 IFA$="Y" THEN 1600
2500 IFA$<>"N"THEN2460
2520 S$="!C!E!C(W<R<"+CHR$(198):K=USR(0):GOSUB4860
2540 PRINT@PO,CS;
2560 FORX=1TO3
2580 FORY=R1(M(X))TO1STEP-1
2600 IFL(NP,M(X),R1(M(X)))=NPTHENPRINT@P(M(X),Y),CB;:V(M(X))=3:S
$="("+CHR$(50-3*Y):K=USR(0):GOTO2660
2620 IFL(NP,M(X),Y)=5THENL(NP,M(X),Y)=NP:PRINT@P(M(X),Y),T$(NP);
:GOTO2660
2640 IFL(NP,M(X),Y)=NPTHENL(NP,M(X),Y)=0:PRINT@P(M(X),Y)-1,B$;CH
R$(149);
2660 NEXTY
2680 IFL(NP,M(X),R1(M(X)))=NPTHENW(NP)=W(NP)+1
2700 NEXT X
2720 IFW(NP)>2THEN2800
2740 FORX=2TO12:IFV(X)=2THENV(X)=0
```

```
2760 NEXTX
2780 GOTO1360
```

A player has won.

```
2800 PRINT@PO,CS"    W I N N E R  * * *   "N$(NP)"  * * *  W
 I N N E R"
```

Play winning music.

```
2810 FORK=1TO5:S$=AZ$(K):FORU=1TO10:AZ=USR(0):NEXT:NEXT
```

Prompt for next game.

```
2990 PRINT"HEY, THAT WAS FUN! ";:FORTIM=1TO1000:NEXT:PRINT"WANT
TO DO IT AGAIN?";
2992 I$=INKEY$:IFI$=""THEN2992ELSEIFI$="Y"THENRUN
2995 GOSUB3160:CLS:CLEAR50:END
```

Program logic and control subroutines.

```
3020 PS=POS(0)-17:FORX=1TO6
3040 PRINT@PO,CHR$(191+PS);:PRINT@PO,"    ILLEGAL CHOICE!";
3060 FORTIM=1TO200:NEXT
3080 PRINT@PO,CHR$(191+PS);
3100 FORTIM=1TO200:NEXT
3120 NEXTX
3140 RETURN
3160 S$="<C(W(W<R<W":K=USR(0)
3180 FORTIM=1TO300:NEXT
3220 RETURN
3260 '
```

What to PRINT after token.

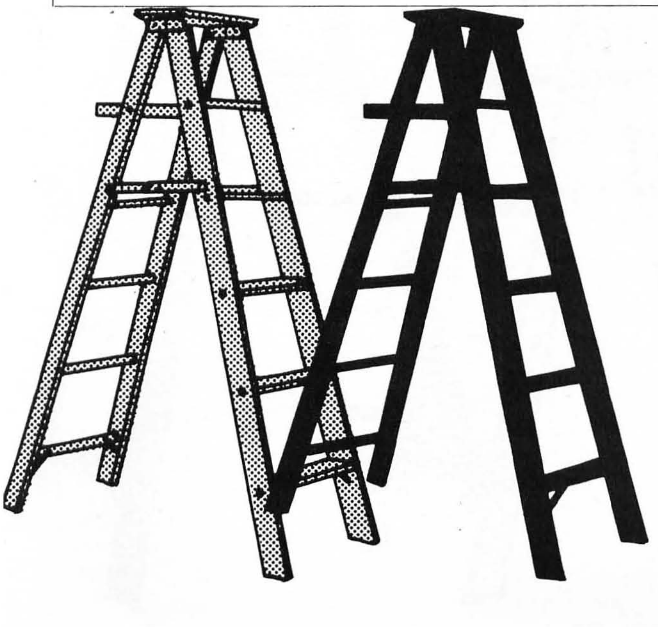
```
3300 NZ=P(S(Z),U)/64:S$="("+CHR$(NZ*3):K=USR(0)
3320 PRINT@(P(S(Z),U)-1),B$;CHR$(149);
3340 FORW=1TOTP
3360 IFL(W,S(Z),U)>0ANDL(W,S(Z),U)<5THENPRINT@P(S(Z),U),T$(W);
3380 NEXTW
3400 RETURN
3440 A$=INKEY$:IFA$=""THEN3440ELSERETURN
3460 '
```

Player blew it.

```
3480 PRINT@PO,CS,"   You blew it, "N$(NP)"!!!"
3500 GOSUB3160
3550 GOSUB4860
3560 FORX=1TO3
3580 FORY=R1(M(X))TO1STEP-1
3600 IFL(NP,M(X),Y)=5THENL(NP,M(X),Y)=0:V(M(X))=0:GOTO3640
3620 GOTO3880
3640 '
3660 F4=0:IFR1(M(X))=YTHENF4=Y
3680 PA=P(M(X),Y):C1=CHR$(130):C2=CHR$(129):C3=CHR$(132):C4=CHR$
(136)
3700 FORX1=1TO7
3720 PRINT@PA,"  "C1;:PRINT@PA,"  "C2;:PRINT@PA," "C1" ";:PRINT@
PA," "C2" ";:PRINT@PA,C1"  ";:PRINT@PA,C2"  ";:PRINT@PA,C3;:PRIN
T@PA,C4;:PRINT@PA," "C3;:PRINT@PA," "C4;:PRINT@PA,"  "C3;:PRINT@
PA,"  "C4;
3760 NEXTX1
3780 IFF4>0THENPRINT@P(M(X),Y),M(X);ELSEPRINT@P(M(X),Y),STRING$(
3,95);
3820 FORW=1TOTP
3840 IFL(W,M(X),Y)>0ANDL(W,M(X),Y)<5THENPRINT@P(M(X),Y),T$(L(W,M
(X),Y));
3860 NEXTW
3880 NEXTY
3900 NEXTX
3920 FORX=2TO12:IFV(X)=2THENV(X)=0
3940 NEXTX
3960 GOTO1360
4020 '
```

Instructions.

```
4040 PRINT@PO,CS;Q$"LADDERS"Q$"  is a dice game played by 2 to 4
 people.
 OBJECT:  to reach the top of any three ladders.";:GOSUB4880
```



```
4080 PRINT" To start your turn, roll all four dice. Split the ro
ll in half
in any way you wish, and add the two dice in each half to create
a pair of numbers.";:GOSUB4880
4100 PRINT"EXAMPLE:  With this roll, you can create any of the p
airs:
$ 6 - 10:  (1+5) and (4+6) $   $ 5 - 11:  (1+4) and (5+6) $
$ 9 - 7:  (5+4) and (1+6)";
4120 FORTIM=1TO500:NEXT
4140 PRINT@PD,D$(1);:PRINT@PD+128,D$(5);:PRINT@PD+256,D$(4);:PRI
NT@PD+384,D$(6);:GOSUB4880
4160 PRINT" The pair of numbers you choose to create represents
the two
ladders onto which you must now place one of your 3 available
markers. Say you choose as a pair  6 and 10";:GOSUB4880
4180 PRINT"
You must now place a marker onto the bottom rung of ladders
"Q$"6"Q$" and "Q$"10"Q$".";:GOSUB4900:PRINT@R(6),T$(5);:PRINT@R(
10),T$(5);:GOSUB4880
4200 PRINT"  In this game you may roll more than once on a singl
e turn.
On each additional roll, you also create a pair of numbers in
the same way.";:GOSUB4880
4220 PRINT"Let's say you roll again and create a pair that inclu
des a
number you've already chosen. When this happens, the marker
moves up one rung on that number's ladder.";:GOSUB4880
4240 PRINT"Let's say you roll again and decide to create a pair
with a NEW
number. If you have another marker left, you MUST place it onto
the new marker's ladder.";:GOSUB4880
4260 PRINT"EXAMPLE:  Rolling on your same turn say you get this:
";
4280 GOSUB4860
4300 FORTIM=1TO1000:NEXT:PRINT@PD,D$(2);:PRINT@PD+128,D$(4);:PRI
NT@PD+256,D$(3);:PRINT@PD+384,D$(5);:FORTIM=1TO1000:NEXT
4320 PRINT@PO,"If you choose 6 and 8 as your pair, the marker on
 ladder "Q$"6"Q$"
moves up one rung and the last marker goes on "Q$"8"Q$".";
4340 GOSUB4900:PRINT@R(6),CC;:PRINT@R(6)-64,T$(5);:FORTIM=1TO750
:NEXT:PRINT@R(8),T$(5);:GOSUB4880
4360 PRINT"If, instead, you choose to create 5 and 9 as your pai
r, you
must place your last marker onto EITHER of these ladders and
ignore the other ladder.";
4380 PRINT@R(6)-64,CC;:PRINT@R(8),CC;:PRINT@R(6),T$(5);:PRINT@R(
8),CC;:GOSUB4900
4400 FORX=1TO3
4420 PRINT@R(5),T$(5);:PRINT@R(9),CC;
4440 FORTIM=1TO1250:NEXT:PRINT@R(9),T$(5);:PRINT@R(5),CC;
4460 FORTIM=1TO1250:NEXT
4480 NEXT X:PRINT@R(9),CC;:GOSUB4880
4500 PRINT"
If you choose to create 7 and 7 as your pair, the last marker
moves up two on the "Q$"7"Q$" ladder.";:GOSUB4900
4520 PRINT@R(7),T$(5);:FORTIM=1TO500:NEXT:PRINT@R(7)-64,T$(5);:P
RINT@R(7),CC;
4540 PRINT@PO-64,"You may continue to roll as long as your last
roll allowed you
either to place one of your three markers or to move one up. If
you prefer, you MAY STOP WHENEVER YOU WISH.";:GOSUB4880
4560 PRINT"
When you stop, your personal token will be substituted for the
marker, and the dice pass to the next player.";
4580 GOSUB4900:N$="NAM":PRINT@R(7)-64,N$;:PRINT@R(6),N$;:PRINT@R
(10),N$;:GOSUB4880
4600 PRINT"
On preceeding turns, if you choose a ladder with one of your
tokens on it, the marker goes on the next rung up.";:GOSUB4880
4620 PRINT"When your roll WILL NOT allow your either to place a
marker or
to move one up, you've "Q$"BLOWN IT"Q$". Your MARKERS are remove
d and
your turn ends. Your NAME TOKENS remain, however.";:GOSUB4880
4640 PRINT"
You win a ladder if you end your turn with a name token covering
the number at the top of the ladder.";:GOSUB4900
4660 PA=R(10)-320:PRINT@PA-64,N$;:FORTIM=1TO750:NEXT
4680 FORX=PATOR(10)STEP64
4700 PRINT@X,CB;
```

```
4720 S$="("+CHR$(70+X/64*3):K=USR(0)
4740 NEXTX
4760 GOSUB4880
4780 PRINT"You may not place a marker on a ladder that someone h
as already
won - EVEN IF YOU'VE WON THAT LADDER YOURSELF. The winner is
the first player to win any three ladders.";:GOSUB4880
4800 N$=STRING$(3,95):PRINT@R(6),N$;:PRINT@R(7)-64,N$;
4820 PRINT@PA-64," 10";:FORX=PATOR(10)STEP64
4840 PRINT@X,N$;:NEXT:S$="(Z(N(ZP2PM":GOSUB4860:GOTO0880
4860 FORX=PD-2TOPD+446STEP64:PRINT@X,"          ";:NEXT:RETURN
4880 PRINT@1017,"More ?";:GOSUB3440:PRINT@PO-64,CS;:RETURN
4900 PRINT@1017,"More ?";:GOSUB3440:RETURN
4920 '
```

PRINT game board.

```
4940 CLS:POKEE-105,1:POKEE-91,2
4960 PRINT@PO,CS,"HERE IS THE "Q$"LADDER BOARD"Q$;
4980 ST=-1:DD=2
5000 ST=ST+DD
5020 PB=PB+64-DD*2
5040 N1=N1-1:N2=N2+1
5060 IFDD>1THENPRINT@PB-63,N1;:GOTO5100
5080 R(Z)=PB+1:R(14-Z)=PB+Z1-2:Z1=Z1-8:Z=Z+1
5100 PRINT@PB,;
5120 FORX=1TOST
5140 PRINTB$;
5160 NEXT
5180 PRINTCHR$(149);:IFN2<13THENPRINTN2;
5200 IFST=11ANDDD>1THENST=13:DD=-DD:PB=PB-4
5220 IFPB>768THEN5260
5240 GOTO5000
5260 DD=2:Z=3
5280 FORY=2TO12
5300 PA=R(Y)
5320 FORX=0TOZ
5340 P(Y,X+1)=PA-X*64
5360 NEXTX
5380 R1(Y)=Z
5400 Z=Z+DD:IFZ=13THENDD=-DD
5420 NEXTY
5440 RETURN
5460 PQ=0:PU=5
5480 PRINT@0,"IV - LM - M";
5500 FORX=2TO12
5520 IFV(X)=2THENPQ=PQ+64:PRINT@PQ,X;
5540 IFV(X)=1THENPU=PU+64:PRINT@PU,X;
5560 NEXT X
5580 PRINT@74,M;
5600 RETURN
5620 '
5640 READ D
5660 FORX=1TO6
5680 IFI>2THEN5720
5700 IFD=DS(X)THENI=I+1:S(I)=DS(X)
5720 NEXT X
5740 IFI<3THEN5640
5760 RETURN
```

Data for lower-case (HHH), sound (MMM) and winning music (WWW).

```
5780 DATA HHH,221,110,3,221,102,4,218,154,4,221,126,5,183
5800 DATA 40,1,119,121,254,32,218,6,5,254,128,210,166,4
5820 DATA 195,125,4,82,105,107,0,80,105,101,114,99,101,33
5880 DATA MMM,46,115,38,126,126,183,245,35,94,35,86,235
5900 DATA 241,200,61,200,61,245,86,30,1,29,35,78,35,62,9
5920 DATA 211,255,65,27,122,179,40,2,16,249,40,229,62,11
5940 DATA 211,255,65,27,122,179,40,2,16,249,32,228,24,213
5960 DATA 211,255,65,27,122,179,16,251,201,WWW,50,100,200
5970 DATA 50,120,180,60,120,180,30,60,90,25,50,100
```

Programmer's routine to test the sound generator.

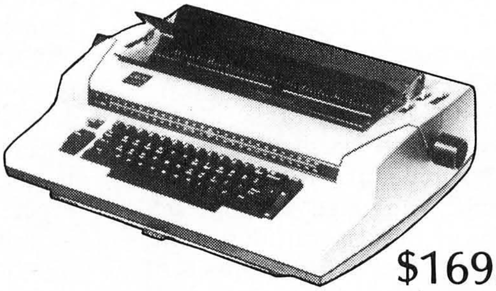
```
5980 PRINTA$" ";
6000 A$=INKEY$:IFA$=""THEN6000
6020 S$="("+A$
6040 K=USR(0)
6060 GOTO5980
```

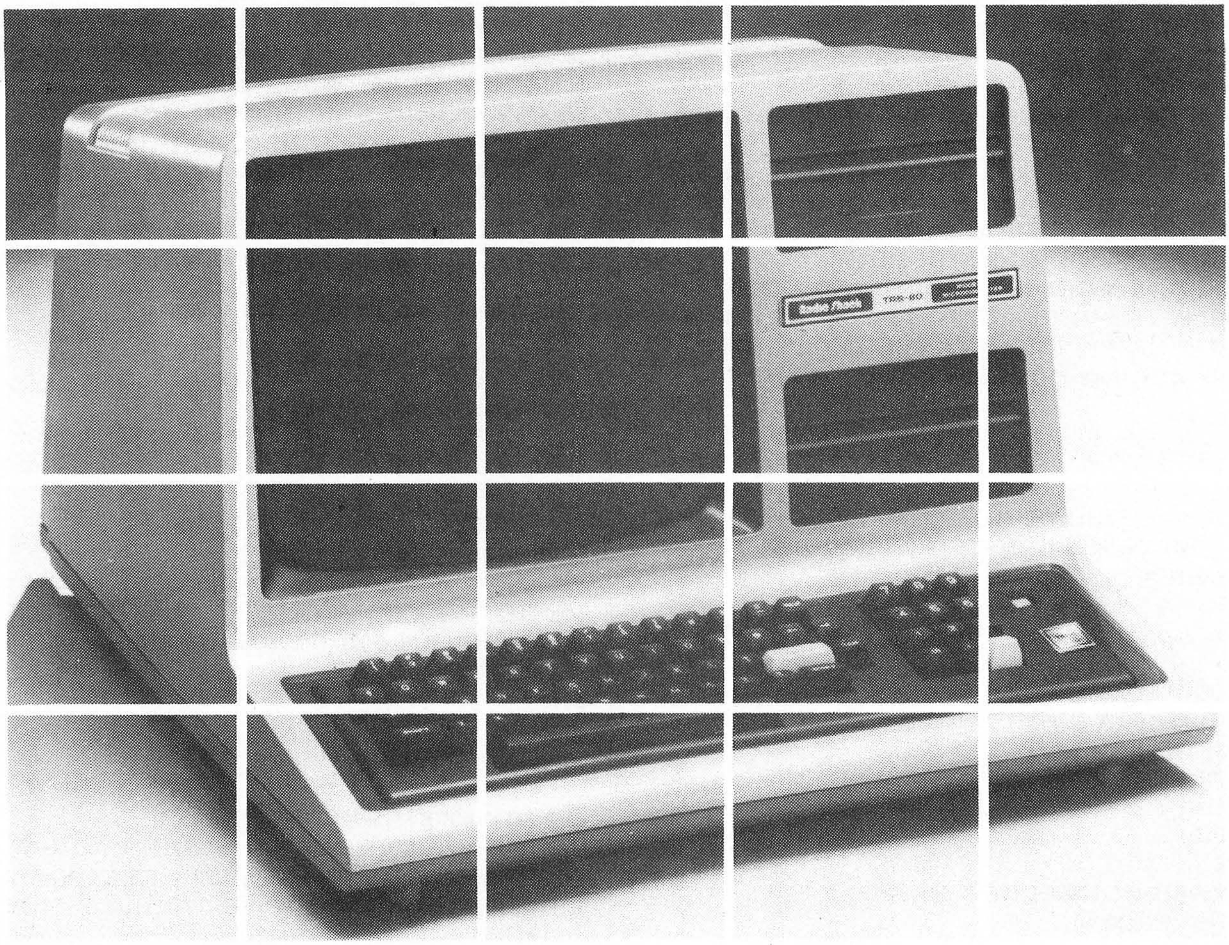
Another sound test routine.

```
6080 INPUT"USER NUMBER";U
6100 S$="!"+CHR$(U):FORX=1TO50:K=USR(U):NEXT
6120 GOTO6080
6140 '
```

Set up sound strings.

```
6160 S1$="(W(P(WP'":S2$="(W(P(WP"+CHR$(218)
6180 PS$="(J+7(+<$"+CHR$(20)+"$($<+"+CHR$(20)+"+(+(7(+(7PJ"
6720 READA$:IFA$<>"WWW"THEN6720ELSEFORAZ=1TO5:AZ$(AZ)="":FORU=1T
O3:READK:AZ$(AZ)=AZ$(AZ)+CHR$(7)+CHR$(K):NEXT:NEXT:RESTORE
6740 RETURN
```





NEWDOS-80 ENHANCEMENTS TO SEQUENTIAL DATABASE

by Robert Jacobs

Users of some of the new operating systems such as LDOS, NEWDOS-80, and DOSPLUS have the opportunity to use the additional features of their system to advantage in programs like *SoftSide*'s *Database*. The BASIC enhancements of NEWDOS-80 particularly lend themselves to such use. The availability of low-memory Machine Language sorting, access to the disk operating system, and the new file types of NEWDOS-80 offer greater flexibility and speed. Some enhancements of the *SoftSide Database* for use with NEWDOS-80 follow; note that the sort routine requires Version 2.0.

Among the most important developments of NEWDOS, though least discussed in the literature thus far, are the five new BASIC file types. One of these, the "MU" file offers a number of advantages to users and developers of *SoftSide*'s *Database.* Unlike TRSDOS sequential files, the MU file can be partially updated and randomly accessed. It may be extended without rewriting the entire file. It also requires less disk space, especially for numeric data, and is more easily programmed once its syntax is understood. This suggests a whole new realm of possibilities for database users.

Happily, the initial steps are much simpler than the specifications of the NEWDOS-80 manual suggest. The conversion to the MU file type requires that only a few lines of the *SoftSide Database* be altered; line numbers given are valid for both the original version of September 1980 and the present TRS-80® translation.

```
1010 OPEN"I",1,F$,"MU"
1020 GET 1,,,NH,NI;
1140 FOR I=0 TO NH:GET 1,,,H$(I);:NEXT
1260 GET 1,,,I$(I,J);
2050 OPEN"O",1,F$,"MU"
2060 PUT 1,,,NH,NI;
2080 PUT 1,,,H$(I);
2250 PUT 1,,,I$(I,J);
```

The database, enhanced as above, will read and write MU files; routines to access these files for partial record input and output can be written by using NEWDOS-80's OPEN"R" or OPEN"D" functions. For the details of such use, Apparat's manual should be consulted.

Existing data files may be very easily converted to MU format by using the file conversion program contained in Figure 1. This program reads in the whole file and then writes it back out as a MU file under the same or a different filename. If experimentation with this form of file interests you, I recommend that you establish separate working copies of the database and your data files, perhaps using the extension /MU, until you are completely satisfied that your program is doing what you want it to do.

FILECONV/BAS, as I call it, is quite straightforward. It prompts you to enter the old and new filespecs and then performs the conversion. If your TRS-80® has less than 48K you will have to reduce the amount of string space cleared in line 10. Should you wish to place your MU files on separate disks from your convential files, the following line may be added to FILECONV/BAS:

```
135 CLS:PRINT"MOUNT DESTINATION DISK AND PRESS (ENTER) WHEN READ
Y";:GOSUB 60000
```

Incidentally, this conversion program is readily adaptable to any sequential access file: Dimension your arrays, read in the variables just as you did under TRSDOS, and write them back out again in the MU file format. Thus you will be able to take advantage of the new file structures without lengthy retyping of data.

Figure 2 replaces the entire BASIC sort routine as it appears in the database. This program fragment allows one to take advantage of the low-memory machine sort of NEWDOS-80. New arrays are dimensioned in Line 7100. Q%(I) becomes an independent integer array which carries the sorted order after either line 7110 or 7120, depending on whether the sort is ascending or descending. This array is used in turn to establish a three-dimensional array, V%(I,J,2), carrying sorted VARPTR information for the main data array.

Then, in lines 7160 to 7168, the variable pointers are POKEd back into memory, thus rearranging the data without actually moving strings. There is no garbage collection, and consequently no long sorting delay. With the modified database I have sorted an array involving 70 long string items, seven headings deep, in less than three minutes.

Note the use of CMD"F= ERASE" in line 7170. This entirely clears out the temporary arrays Q%(I) and V%(I,J,K) which may help the remainder of the program to operate a little faster, although if sorting is to be done there must still be enough non-string space to permit the establishment of these temporary arrays. The erasure also prevents redimension errors if the sort is accessed again in the same run of the program.

Variables

File Conversion Program:

A$: User answers.
F1$: Filename of file to be converted.
F2$: New file filespec.
H$(): Heading string array.
I,J: Loop counters.
NH: Number of headings
NI: Number of items

Figure 1: FILECONV/BAS

```
10 CLEAR24000:CLS:PRINT"FILE CONVERSION, SOFTDATA FORMAT TO NEWD
OS-80 'MU' FORMAT"
20 PRINT:LINEINPUT"NAME OF FILE TO BE CONVERTED  ";F1$
30 PRINT:LINEINPUT"NEW FILE NAME  ";F2$:PRINT
40 IFF2$=F1$ THENPRINT"OLD FILE WILL BE DESTROYED; PRESS <A> TO
APPROVE,
ANY OTHER KEY TO CHANGE OUTPUT FILENAME.":GOSUB60000ELSE60
50 IFA$<>"A"THEN30
60 CLS:PRINT"CHECK DISKS ONCE MORE - THEN PRESS <ENTER>.":GOSUB6
0000
70 OPEN"I",1,F1$
80 INPUT#1,NH,NI
90 DIMH$(NH),I$(NI,NH)
100 FORJ=0TONH:INPUT#1,H$(J):NEXT
110 FORI=0TONI:FORJ=0TONH
120 INPUT#1,I$(I,J):NEXTJ,I
130 CLOSE
140 CLS:PRINT"WRITING NEW FILE"
150 OPEN"O",1,F2$,"MU"
160 PUT 1,,,NH,NI;
170 FORJ=0TONH:PUT 1,,,H$(J);
175 NEXT
180 FORI=0TONI:FORJ=0TONH
190 PUT 1,,,I$(I,J);
200 NEXTJ,I:CLOSE
210 CLS:PRINT"CONVERSION COMPLETE.  ANOTHER FILE (Y/N)?":GOSUB60
000
220 IFA$="Y"THENRUNELSEEND
60000 A$=INKEY$:IFA$=""THEN60000ELSERETURN
```

Variables

Sort Routine for NEWDOS-80 2.0:

I, J, K: Loop counters for VARPTR array.
Q%(I): Integer array; holds sorted order after line 7100 or 7200.
V%(I,J,2): VARPTR information for main data array.

Figure 2: SORT 80

```
7100 DIMV%(MX,NH,2),Q%(MX):ONAGOTO7110,7120
7110 CMD"O",NI+1,*Q%(0),I$(0,J1):GOTO7130
7120 CMD"O",NI+1,*Q%(0),-I$(0,J1)
7130 FORI=0TONI:FORJ=0TONH:FORK=0TO2
7140 V%(I,J,K)=PEEK(VARPTR(I$(Q%(I)-(MX*J1)-J1,J))+K)
7150 NEXTK,J,I
7160 FOR I=0TONI:FORJ=0TONH:FORK=0TO2
7165 POKE VARPTR(I$(I,J))+K,V%(I,J,K)
7168 NEXTK,J,I
7170 CMD"F=ERASE",V%(0,0,0),Q%(0)
7200 SS=0:RETURN
```

Robot Attack

A review by Michael McKenna

From Big Five Software, P.O. Box 9078-185, Van Nuys, CA 91409. A TRS-80® 16K cassette or 32K disk program. Suggested retail price: Cassette — $15.95, disk — $19.95.

The robots are getting closer and closer. You're surrounded! You shoot one and it's gone. There, another one's destroyed! Wait! The robot in front of you is shooting! You quickly move down, but alas, the robot's deadly laser blast has vaporized your body.

> “. . . *Robot Attack* is fun, challenging, terribly frustrating, and WELL worth the money.”

A Long Time Ago . . .

All of the above takes place a long time ago and in a galaxy far, far away, as you attempt to invade and recover a space station that has been taken over by robots. The robots have taken over the space station and use it to fuel and repair their Jidyan Land Kruisers (sound familiar, *Super Nova* owners?). You have been chosen for the secret mission to take the station. The mission's code name is Robot Attack.

Most of the preceding paragraph is introduced to you at the beginning of the program using big graphics letters that slowly scroll up the screen in a *Micro Marquee*-like style.

Talking?

Yes. As the ad says, in addition to excellent sound effects, *Robot Attack* talks! You need no special hardware; the voice, along with the sound effects, comes through the cassette port. The voice effects aren't voice synthesizer quality, but it is very clear what it is being said.

The cassette version of *Robot Attack* announces its name, "player one," "player two," "chicken," "lucky," "great score," and the inevitable "game over." The disk version says everything the cassette version does but it replaces "lucky" with "that was pure luck," and if you get the highest score so far, it spiels out, "You have achieved the highest score so far. Enter your initials carefully, so that we may be able to save them permanently." The disk version uses about twice as much data and the quality of the voice is improved quite a bit.

The Game

If you are familiar with the arcade game *Berzerk*, then you can have a good idea of what *Robot Attack* is like. You are transformed into a little graphics character which is placed inside a simple maze filled with moving, graphics robots. There are exits on the top, bottom, and both sides. The object is to shoot all of the robots in the room without touching a robot, an electric wall, or a laser shot, then leave through an exit (although you can leave before all robots have been destroyed). You start with four humanoids, and are awarded a new one with each 5000 points you get. The game ends when you lose all of your humanoids.

You can steer the humanoid around the room by using the arrow keys, the I, J, K, and M keys, or the joystick available from Big Five or Alpha Products Co. To shoot, hold down the space bar (or the F key or the button on the joystick) and then press the arrow(s) that point in the direction you want to shoot. You can have up to three shots in the air at one time.

If a robot is hit by a laser or runs into a wall or another robot, it is obliterated and you are awarded 50 points. If you want a bonus, make sure all of the robots in the room are destroyed, then leave the room by going through an exit. You will hear "lucky" or "that was pure luck," and you will be given a bonus of 40 to 110 points (10 points for each robot that was in the room). If you leave a room before all the robots are destroyed, no bonus is awarded and you are audibly punished with "chicken." Once you exit the room, you're placed in a new one and the exit you used is sealed behind you.

It's not very smart to hang around the room too long, because if you do (drum roll, please) the famous Flagship, which lives in every Big Five game, appears to bless you with its deadly touch! The Flagship beams into the room exactly where you entered it. You are warned that it is coming though, because you can hear and see its flashing transporter beam before it appears.

The Flagship is made from a new "space metal" explains Big Five, and is immune to your lasers, or anything else for that matter! It plows right through walls, robots, and lasers on a collision course that will end in the loss of a humanoid unless you get out of the room. It doesn't bounce like the smiling face in *Berzerk*, however.

After traveling through about five or six rooms, the robots are equipped with lasers which are just as deadly as yours. Now the game is really hard. The robots are pretty good shots, and it's hard to step out of the laser's path in time.

Random Notes

As in all other Big Five games, the top ten high scores, along with the player's name or initials, are kept during the game, and in the disk version, they are permanently saved. You enter your initials or name (whichever fits in four letters) at the close of a game in the big graphics letters used in the rest of the program. Three people's scores are already entered in the high score table, as a built-in feature of the program.

The game can be aborted at any time during play, except when voice effects are going, by pressing BREAK and CLEAR simultaneously. So when things are going bad, just abort the game and forget the whole horrifying experience.

Robot Attack is written in Machine Language and is fast in response time and animation. The sound and graphics qualities are excellent, as in all Big Five games. *Robot Attack* is fun, challenging, terribly frustrating, and WELL worth the money.

A note to joystick users: *Robot Attack* will not work with older versions of the Alpha Product Co. joysticks. If PRINT 255-INP(0) returns a 3 when the button is pressed then *Robot Attack* is not compatible with your joysticks. You can either modify your joystick using instructions provided with the game, or use the keyboard. If PRINT 255-INP(0) returns a 16 while you press the button you're OK.

Missile Attack

A review by Michael McKenna

From Adventure International, Box 3435, Longwood, FL 32750. A TRS-80® 16K cassette or 32K disk program. Suggested retail price: cassette — $14.95, single-density disk — $20.95.

Missile Attack is a fast-paced Machine Language game patterned after Atari's *Missile Command*. The game is written by Philip Oliver of the Cornsoft Group, and is distributed by Adventure International. In *Missile Attack* you are in control of two ABM (Anti Ballistic Missile) launching bases with which you must protect three cities from an unnamed enemy's missiles.

Your two bases are on the bottom left and right. The three cities are evenly spaced between them. As the game starts, the enemy's missiles (represented by graphics lines) begin to fall from the top of the screen. They fall faster and faster as the game progresses. As each missile appears, it adds in its own sound until the onslaught is causing a small roar to come from the speaker. Now it's time to defend yourself.

In order to save your cities you must launch ABMs to intercept the falling missiles. This is done by steering a "+" around the screen using the arrows (or four other optional keys for the PMC-80). Once the "+" is where you would like an ABM to explode (preferably on top of a missile), you press one of the two launch keys. The left base fires when the "@" key is pressed (or another key to match the PMC-80 keys); the right fires when the spacebar is pressed. Then you see and hear your missile rush to the "+" (or where the "+" was — you can move it once the ABM is launched) and explode when it gets there. If the explosion is close enough to the enemy's missile it will detonate it too. For each missile destroyed you get 10 points.

It doesn't pay to miss any missiles because you only have 30 ABMs to destroy each wave which averages about 24 missiles. After each wave is destroyed, you get five points per ABM remaining and 100 points per city remaining. Once your cities are destroyed, they cannot be rebuilt. However, if your bases were destroyed by the attack, they are miraculously rebuilt and restocked with missiles. Then a new wave starts.

If you survive two waves, the value of every point you earn is doubled (i.e., all remaining missiles are worth 10 points each). After two more waves, all values are tripled and so on until the eleventh and twelfth waves, when all point values are worth six times the norm. If you are lucky or skillful enough to survive the twelfth wave of enemy missiles (which are by now really zooming down the screen), you've won.

The game has several extra features such as: keeping the high score, a big graphic "END" when you've been defeated, and a graphics title. The game also has some shortcomings, including running in a continuous loop, i.e., the program goes from title to game to "END" to title to game and so on, with no break! Another problem is that (very rarely) the game will produce an error by keeping some graphics on the screen after a missile has been destroyed. But at the worst, this is only a little distracting.

I really enjoy *Missile Attack*, and find it to be one of my favorite programs. I can win the game by staying alive, unlike many other games that go on forever. Even though I can win almost every time (I've been playing for some time now), I still enjoy it and am always tring for a new high score. *Missile Attack* works on the Model I, Model III and PMC-80, and asks at the beginning which computer you are using. *Missile Command* fans should be warned that *Missile Attack* does not have the advanced features *Missile Command* does, such as splitting missile, planes, killer satellites, etc., but it is still a fun, challenging game.

Microworld

A review by Allen L. Wold

By Arti Haroutunian. From Med Systems Software, P.O. Box 2674, Chapel Hill, NC 27514. For 16K TRS-80® Models I and III. Suggested price: cassette — $19.95, disk — $22.95. Advertised to be available soon for ATARI®.

You are in a white featureless room. In a corner, you see a familiar sight. There is a pocket calculator on the floor. If you figure out how to get to the corner to examine what's there, you find yourself standing in front of a TRS-80® computer. There is a voxbox and a disk drive connected to the computer. The disk drive is empty. Nothing you do to the computer seems to have any effect.

But, if you read the brief instruction manual which comes with the game, you'll know that the calculator is the key to starting the adventure. The puzzle here is so off the wall that only the "clue" (read, explicit instructions) given in the manual will enable you to enter the *Microworld*. Once that's over, you're on your own — more or less.

Microworld is a text adventure game, without graphics. For those who are really "into" the TRS-80®, this is just the thing. Once you have used the calculator to actually start the adventure, you find yourself literally inside a TRS-80®, transformed into an "electroid," traveling through the circuits of the computer. There are over 80 locations to be explored.

The game comes with simple start-up instructions (it is self-loading), and a glossary of technical and pseudo-technical terms, which provide help in solving some of the problems encountered.

For an extra dollar, you can get a hint sheet, which provides help for some of the more difficult problems. This is a good idea, especially since there are few clues in the context of the game to tell you what is expected or what will happen next. The hints are given in a simple substitution cipher, so that you cannot read them accidentally. The cipher key is given, and the deciphering can be done in your head.

The game is quite benign. There is only one instance when the player can be "killed." Usually the worst that can happen is that you get lost in one of the mazes (for which there are no hints), or that the "Charactoid" will come and steal your treasure, which consists of a number of colored IC chips. Then you have to find out where all of them are hidden. There are some locations, however, where by moving on, you suddenly find yourself outside the computer and the game is over. There is so little hazard, that one could spend hours just mapping, without fear of being destroyed. In fact, the first time I played, that's all I did.

The location descriptions are brief, and humorous: "You are in a large room which looks like a military drill field. But you realize you must be in the regulator, since all electrons are being turned into responsible entities." "North Ground Plane. Electrons are constantly falling out of the sky and landing on their valences." "You are in the audio conditioning room. A drill sergeant is trying to get the undisciplined electrons to straighten out, but he is not being very successful." "You are in a 741 opamp. There is a reception desk manned (sic) by a bespectacled female electron. I think she's isotopian."

The problem is that it is unclear at the beginning just what the objective is, and many of the puzzles are indistinguishable from the background environment. For example, in one place it is possible to turn on the disk drives, but the consequences are not given, or even hinted at. There seems to be no change in the environment at all.

There are moments of illogic. One cannot open the large refrigerator, but one can pick it up and carry it along.

However, there is an undeniable educational quality, in that in the course of play, one will become at least superficially familiar with the hardware of a TRS-80® computer. Hobbyists who build their own might find this dull, but for the typical computer owner, and especially for younger players, this is a positive benefit. The intent is not to teach, but the lessons come through anyway.

The puzzles are sometimes quite tricky, especially since so few clues are given in the context of the game. The best strategy seems to be an application of common sense, or, failing that, typical human behavior. A player who is too clever will fail to find the answers to most of the problems presented, but cleverness helps. The instructions tell you that the "Help" command is not strongly supported, and it is not. The problems can be solved, and when you quit, you are given a score out of a possible 145 points. The game may be saved on disk, so that you can start again where you left off.

Exploration is fairly straightforward; you move by compass points or up or down, typing single letters only. Not all exits are given in the location descriptions, of course.

My main problem was that when I met something which looked like a puzzle, I had no idea how to even approach the problem. I'm an experienced *Adventure* player, and this lack of clues bothered me. However, there is little tension in the game, and time passes quickly while you are exploring the circuits of the computer. I would especially recommend this for younger players, and for those who don't like the idea of killing or getting killed.

NEW PRODUCTS

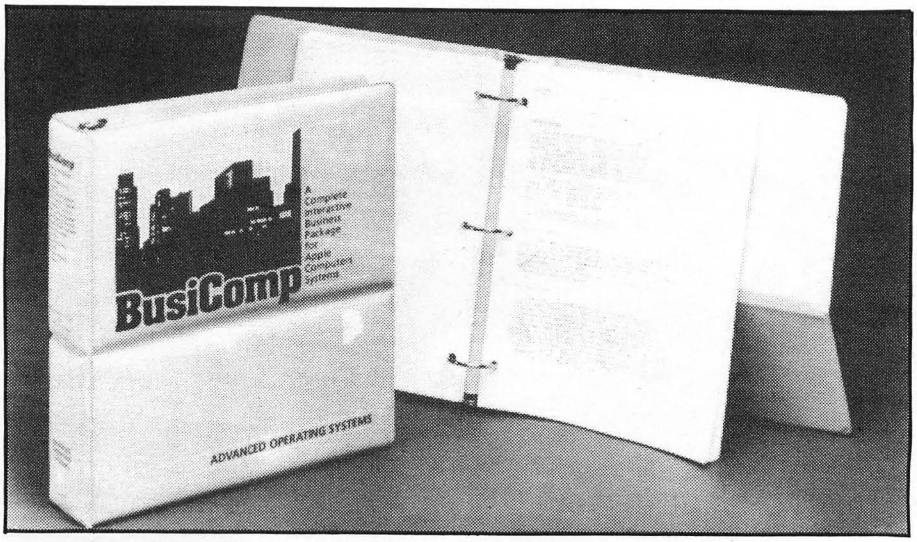
CUESTA SYSTEMS, INC.
3440 Roberto Court
San Luis Obispo, CA 93401

A power supply upgrade package for Apple II and Apple II+ microcomputers called APPLE-UPS lets users take the systems just about anywhere. With APPLE-UPS, the Apple computer can be transformed into a portable information processing system.

The upgrade is available both to sophisticated industrial OEMS and to single users through a Cuesta Systems factory service program. The procedure, which takes about one week, creates a wide and varied range of new applications for the micros, including use in mobile, remote or off-site industrial locations.

The upgrade process revolves around the incorporation of a 12 VDC battery-powered inverter into a standard Apple power supply. Modifications to a circuit board, installation of the inverter module and the addition of a rear-mounted plug with a mating connector are included. A "low-battery" buzzer in the inverter warns when the battery voltage needs recharging.

APPLE-UPS is available immediately, with a list price of $195. Units upgraded by Cuesta Systems at the firm's factory include a nominal shipping and handling charge. Cuesta Systems provides a 90-day warranty on labor and materials supplied in the APPLE-UPS upgrade.


BusiComp
A Complete Interactive Business Package for Apple Computer Systems
ADVANCED OPERATING SYSTEMS

ADVANCED OPERATING SYSTEMS
450 St. John Road
Michigan City, IN 46360
(219) 879-4693

Busicomp, a completely-integrated, interactive business system designed to handle the accounting needs of small businesses, is designed for use on the Apple or Apple II+ computer systems with 48K and at least one disk drive.

Six levels of security open up the 11 program sections which cover accounting functions, including accounts receivable, accounts payable, inventory control, general ledger, fixed assets and payroll. All sections are interactive, allowing simplified, time saving single-step entry.

Busicomp incorporates an excellent error-trapping, menu-driven, self-prompting design which enables the user to begin using the program with limited training.

Among the 41 reports generated by *Busicomp* are aging and data on receivables and payables, personnel, payroll, inventory, expenses, revenue and journal posting. Formats are designed to correspond to standard DSF Forms available nationwide.

Busicomp is available through a select group of retailers. It has a suggested retail price of $1,500.

SYNERGISTIC SOFTWARE
5221 120th Avenue S.E.
Bellevue, WA 98006
(206)226-3216

Over 100 highly-detailed pictures in *Warlock's Revenge* guide you to treasure as well as dangers in this adventure for a 40K ATARI®. As you explore caverns and castles you must use your various skills to obtain the riches therein, while eluding the pitfalls and creatures that abound.

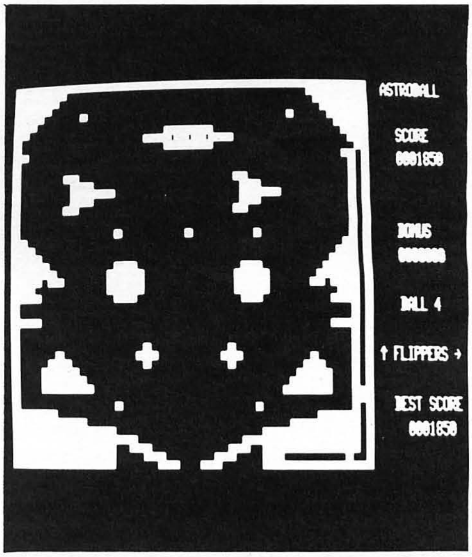
Warlock's Revenge is available on disk for $34.95.

In an effort to inform our readers of new products, ***SoftSide*** welcomes your company's input to this section. Send all information to:
***SoftSide* Publications 6 South Street Milford, NH 03055**

ACORN SOFTWARE INC.
634 North Carolina Ave.,
S.E., Washington, D.C., 20003
(202) 544-4259

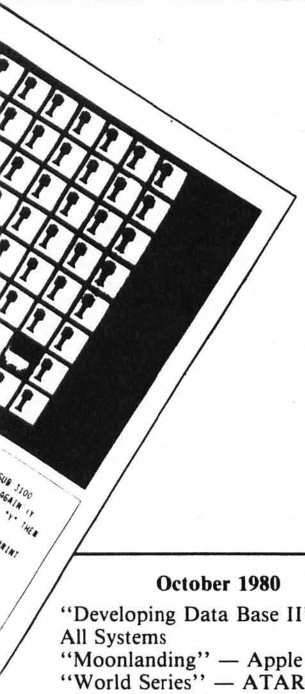
Acorn Software Products, Inc. announces the release of *Astroball* — pinball game with a space theme for the TRS-80® Model I and III. The machine language game, has even more of the high quality sound and graphics associated with its author John Allen. It features space craft, flying saucers and black holes which devour your ball under certain conditions. If you destroy enough meteors, you will get an extra ball. Players are urged to send their high scores to Acorn.

Available in a Model I/III version on 16K tapes or disks for only $19.95 from your software retailer or from Acorn (add $2.00 shipping and handling.)



APPLE CAPTURE by William J. Ryan
Apple Capture is a Hi-Res color graphics game of deduction for two players, requiring Applesoft and 32K RAM.
APPLESOFT BASIC
'APPLE CAPTURE'
AUTHOR: WILLIAM J. RYAN
(C) 1982 SOFTSIDE
Main program.
TRS-80™ Version
SoftSide
Your BASIC Software Magazine • NUMBER ELEV
Battle
Shark
January 1982 Vol. 5, No. 4 THREE DOLLARS
Off and running with GAMBLER
TITAN
November 1981 Vol. 5, No. 2
THREE DOLLARS
Special Issue: Music in the Micro
Flight of the Bumblebee

Back Issues from SoftSide™



VOLUMES FROM THE PAST

If you like what this issue of **SoftSide** has to offer, you should see what's waiting for you in **SoftSide** back issues! You may feel that you've missed out on many of our programs that appeared before you became a subscriber. It's not too late to do something about it. You won't have to miss a thing because we still have back issues available to complete your **SoftSide** library! But, order now as some of our more popular issues are already out of stock and others are dwindling quickly.

Listed below are all of our past issues with their FEATURE programs and the systems they're for. For a more complete index of all the programs and articles offered in each of the back issues of **SoftSide** please refer to the May, 1981, issue. Each issue costs $3.50 for the magazine only. These issues are also available with the programs on cassette for $9.95 or on disk for $14.95 (except DV issues).

The enhanced Disk Versions (DV) contain an extra program for each system. The TRS-80® DV began with the September, 1981, issue. The Apple DV began October, 1981, and the ATARI® DV started in November, 1981. Each enhanced DV costs $19.95.

October 1980
"Developing Data Base II"
All Systems
"Moonlanding" — Apple
"World Series" — ATARI®
"Earth-Port II" — TRS-80®

November 1980
"Developing Data Base III" —
All Systems
"Collision" — Apple
"Trench" — ATARI®
"Kriegspiel" — TRS-80®

December 1980
"Developing Data Base IV"
All Systems
"Baseball" — Apple
"Speedello" — ATARI®
"Kidnapped" — TRS-80®

January 1981
"Developing Data Base V"
All Systems
"Convoy" — Apple
and TRS-80®
"Angle Cannon" — ATARI®
"Ship Destroyer" — TRS-80®

February 1981
"Developing Data Base VI"–
All Systems
"Miner" — All Systems
"Mini-Golf" — ATARI®
and TRS-80®
"Long Distance" — TRS-80®

March 1981
"Developing Data Base
VII" — All Systems
"Strategy Strike" — Apple
and TRS-80®
"Flags" — ATARI®
"Volcano" — TRS-80®

April 1981
"Battle At Sea" — Apple
"Convoy" — ATARI®
"Dominoes" — TRS-80®

May 1981
"Galaxia" — Apple
"Dodge" — ATARI®
"Orienteering At Jacque's
Coulee" — TRS-80®

June 1981
"Old Glory" — All Systems
"Word-Search Puzzle
Generator" — All Systems
"Anallist" — TRS-80®

July 1981
"Chemistry Drill" —
All Systems
"Kidnapped" — Apple
and ATARI®
"Magic Paper
Calculator" — TRS-80®

October 1981
"Leyte" — All Systems
"Developing Data Base"—Apple
"Character Generator" —
ATARI®
"Envyrn™" — TRS-80®

Enhanced Disk Versions
"Super Dairy Farming" — Apple
"Gameplay" — TRS-80®

November 1981
"Flight of the Bumblebee" —
All Systems
"Music Machine" — Apple
"Music Programmer"—ATARI®
"Music Editor" — TRS-80®

Enhanced Disk Versions
"National Anthems" — Apple
"Volleyball" — ATARI®
"Mean Checkers Machine" —
TRS-80®

December 1981
"Titan" — All Systems
"Aircraft Commander" — Apple
"Developing Data
Base" — ATARI®
"Electronics
Assistant" — TRS-80®

Enhanced Disk Versions
"Bobsledding" — Apple
"Survive" — ATARI®
"Konane" — TRS-80®

January 1982
"Gambler" — All Systems
"Microtext 1.1" — All Systems
"Apple Capture" — Apple
"Piazza Hotel" — ATARI®
"TRS-Man" — TRS-80®

Enhanced Disk Versions
"Nuclear Submarine Adventure"–
Apple, TRS-80®
"Death Star" — ATARI®

February 1982
"Space Rescue" — All Systems
"Rubicube" — Apple
"Defense" — ATARI®
"Maze Sweep" — TRS-80®

Enhanced Disk Versions
"Andorra" — Apple
"Kismet II" — ATARI®
"Help Package" — TRS-80®

March 1982
"Hexapawn" — All Systems
"Magical Shape Machine" —
Apple
"Outer Space Attack" —
ATARI®
"Killer Cars" — TRS-80®

Enhanced Disk Version
"PEEKER/POKER" — Apple
"Curse of the Pharaoh" —
ATARI®
"Warpath" — TRS-80®

April 1982
"Microtext" — All Systems
"Poster Maker" — Apple
"ATARI® Banner
Machine" — ATARI®
"Database" — TRS-80®

Enhanced Disk Versions
"Semaphore" — Apple
"Renumbering for the
ATARI®" — ATARI®
"Screen Print" — TRS-80®

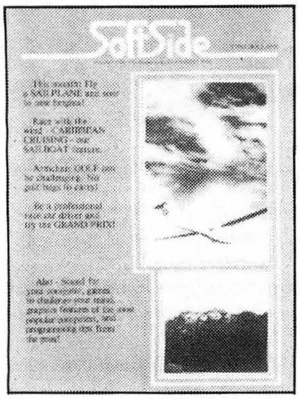
Use card provided in this issue to order or send a list of the back issues you'd like, with payment of $3.50 per magazine (minimum order of 3 issues) to:

SoftSide Publications
515 Abbott Drive
Broomall, PA 19008

To order the magazine/media combinations, use the card provided or send $9.95 per cassette and magazine, $14.95 per disk and magazine, or $19.95 per DV and magazine to:

SoftSide Publications
6 South Street
Milford, NH 03055

Prices for USA orders only. For foreign orders, see page 16.



Back Issue of the Month: August 1980

A historic issue: We went to a full-size magazine format and added the ATARI® to our lineup of systems.

The TRS-80® programs are in a sporting vein: *Sailplane, Pro-Tour 80,* and *Grand Prix* will keep you soaring, putting, and racing for many hours. Or, take your Apple *Caribbean Cruising*, play a cool game of Concentration, and round off the trip with a lesson in animating *ROM the Robot*. If you prefer the company of an ATARI®, you can choose a relaxing afternoon of *Master's Golf*, a tense *Maze Search* for lurking demons, or a last-ditch effort to save earth in *Stratoblaster Outpost.*

Article topics include sound generation and comparative graphics on all three systems. There's even a full-page ad for a year's subscription to ***SoftSide*** for $15.00, and for DV at $4.00 MORE than the current rate!

While our supply lasts, this classic issue is available for a mere $3.50. See our Back Issues ad and page 16 in this issue for ordering information.

Advertiser's Index

MACHINE HEAD
BY SPYDER
WILL YOU PLEASE QUIT YOUR DAYDREAMING? YOU'VE BEEN SLOWER'N A TURTLE ALL WEEK!
WHAT'S WRONG WITH YOU?
IT'S SPRING, MACHINE HEAD!
"FOR, LO! THE WINTER IS PAST, THE RAIN IS OVER AND GONE;
"THE FLOWERS APPEAR ON THE EARTH;
"THE TIME OF THE SINGING OF BIRDS IS COME..."
SPRING IS THE SEASON OF RENEWAL AND REBIRTH, MACHINE HEAD!
BUT THEN, BEING A MACHINE, YOU WOULDN'T KNOW ABOUT ANY OF THAT
I KNOW THAT IF YOU DON'T RENEW YOUR WORK, YOU'LL BE REBORN UNEMPLOYED
5-82
SPYDER

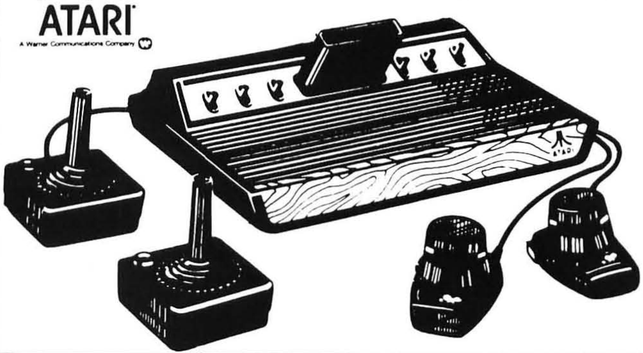
VIDEO COMPUTER SYSTEM

● Create excitement with your home TV. Sharpen mental and physical coordination with a full library of challenging, sophisticated video games. ● Bright, crisp color (on color TV). ● Realistic sound effects (through the TV speaker). ● Interchangeable controllers and cartridges to enjoy the full range of **Video Computer System** games. ● Difficulty option switches so the games get harder as you get better. ● Special circuits to protect TV. ● System comes with Combat Cartridge, Joystick and Paddle Controller, AC adaptor and TV switch box. ● Similar to coin operated games. ● Used with player's own TV set.

SUGGESTED RETAIL $199.95

ATARI
A Warner Communications Company

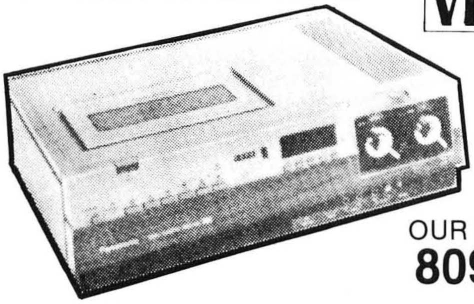
OUR PRICE
$179.99



- ■ Asteroids
- ■ Warlords
- ■ Breakout
- ■ Superman
- ■ Home Run
- ■ Bowling
- ■ Canyon Bomber
- ■ Backgammon
- ■ Championship Soccer
- ■ Casino
- ■ Space Invaders
- ■ Video Pinball
- ■ Adventure
- ■ Night Driver
- ■ Circus Atari
- ■ Maze Craze
- ■ Football
- ■ Video Checkers
- ■ Dodge 'Em
- ■ Slot Racers
- ■ Missile Command
- ■ Othello
- ■ Basketball
- ■ Game Cartridge Storage Unit
- ■ Indy 500
- ■ Video Chess
- ■ Video Olympics
- ■ Air-Sea Battle
- ■ Tic-Tac-Toe
- ■ Golf

2/4/6 HOUR VIDEO TAPE

VHS™



OUR PRICE
809.99

SUGGESTED RETAIL $899.95

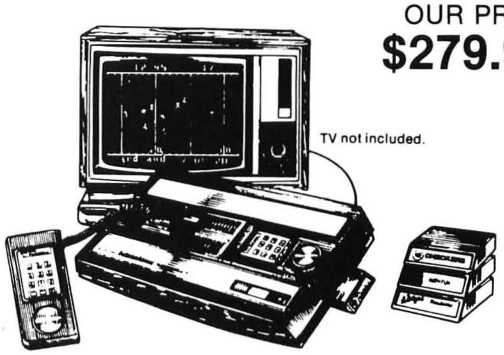
● New 6-hour "Super Long Play" video tape recorder with soft touch controls. ● Switchable to "Long Play" 4 hour and Standard 2 hour recording. ● Compatible with other 2/4 and 2/4/6 hour VHS systems. ● Record programs off the air or record home movies with an optional TV camera. ● Automatic tape threading. ● Hot pressed ferrite video heads. ● Direct drive video head cylinder motor. ● Azimuth recording system. ● Built-in UHF and VHF tuner allows recording one station while viewing another. ● Tracking control. ● 100/ solid state integrated circuitry. ● Pause mode; complete with remote pause control. ● Auto-digital memory counter provides automatic stop while rewinding. ● Complete with all accessories for hookup to all types of TV sets and antennas.

INTELLIVISION

● The video television system which provides an endless variety of game play and involvement. ● Compatible with any color TV set. ● Utilizes a full selection of Mattel game and learning cartridges. ● Each cartridge comes with overlays which fit directly over the two individual hand controllers for easy game play decisions (no additional controllers). ● Includes Las Vegas Poker/Blackjack cartridge. ● Objects of the game screen can be maneuvered in 16 different directions by the object control disc. ● 16-bit microprocessor provides a full range of sound effects, music, color and an extraordinary high level of resolution.

SUGGESTED RETAIL $299.95

OUR PRICE
$279.99



TV not included.

MATTEL ELECTRONICS

- ■ Astrosmash
- ■ Space Armada
- ■ NASL Soccer
- ■ Auto Racing
- ■ Adventure
- ■ Boxing
- ■ NHL Hockey
- ■ Horse Racing
- ■ Space Battle
- ■ NFL Football
- ■ Tennis
- ■ Las Vegas Roulette
- ■ Sea Battle
- ■ NBA Basketball
- ■ PGA Golf
- ■ Backgammon
- ■ Armor Battle
- ■ Major League Baseball
- ■ U.S. Team Skiing
- ■ Checkers

TSE 14 South St., Milford, N.H. 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1760

master charge THE INTERBANK CARD   VISA®   AMERICAN EXPRESS

STOP TYPING!

Get Instant Enjoyment from SoftSide's programs with SoftSide's Cassette Version (CV) and Disk Version (DV)!

Our media editions let you spend less time TYPING — and more time USING the fine software that **SoftSide** brings you every month. And we let you choose the version you want.

Cassette Version (CV)

SoftSide's Cassette Version (CV) offers you an inexpensive way to enjoy our programs without hours of typing or hunting for errors. All programs are tested and ready to go!

CV gives you the programs offered for your system each month in **SoftSide** on a tape, plus the magazine itself — 12 magazines and 12 tapes per year for just $75.

Disk Version (DV)

DV contains a BONUS program for your system on the disk in addition to the other programs available that month. Only the documentation for the bonus programs will appear in **SoftSide** magazine, NOT the code. The bonus programs will be of every conceivable type — multiple and Machine Language programs, modified languages, ongoing modular programs and software so extensive, it would take an entire issue of **SoftSide** just to print the code.

Feel like you're missing something? You are. Don't wait to take advantage of our offer — 12 magazines and 12 disks for just $125 a year. For orders outside the U.S., add $50. For your convenience we also offer an installment payment plan for MasterCard and VISA holders: Pay just $32.50 per quarter (a total of $130 which includes a $5 billing charge).

To order, use the card provided in this issue.