

\$1.50

Washington Apple Pi



Volume 3

May 1981

Number 5

Highlights

REVIEWS OF THE PASCAL LIBRARY
DISKS

SINGLE DRIVE CONVERT

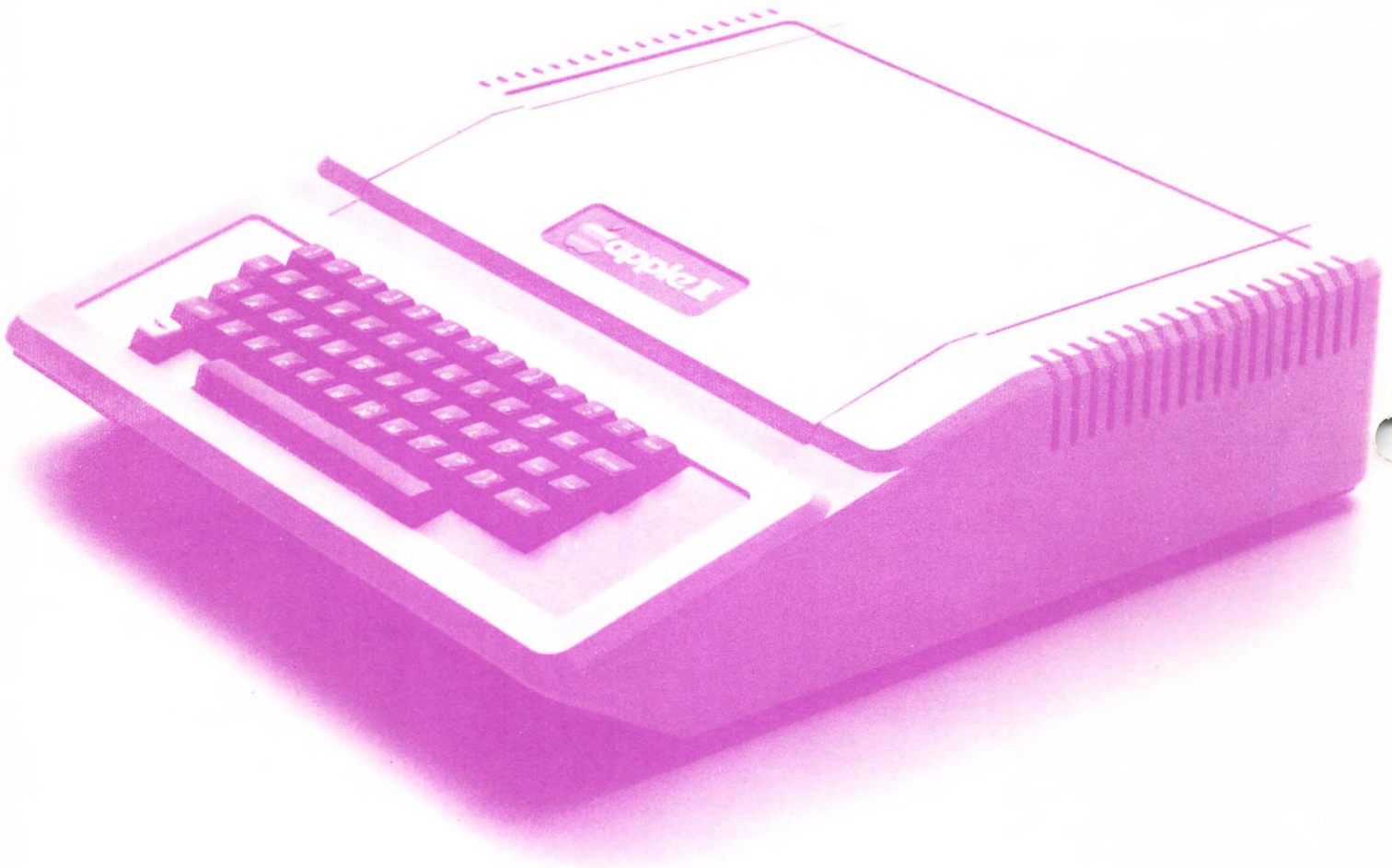
BLAISE AWAY

In This Issue

	Page
MEMBERSHIP INFORMATION, EVENT QUEUE, EDITORIAL	1
UPDATE NOTES ON THE MX-80. PETER ROSDEN	1
MINUTES, NOTICES	2
SIGNEWS.	2
APPLE FOR THE ELDERLY OR HOPE FOR CHANGING THE WORLD BERNIE BENSON	4
REVIEW OF THE PASCAL INTEREST GROUP LIBRARY DISKS PAUL A SAND	6
FILE CABINET ENHANCEMENTS. DONALD E KAHLER	9
HYPER HEAD-ON: A REVIEW WALTON FRANCIS	13
SINGLE DRIVE CONVERT - DOS 3.3 TO 3.2. DANA J SCHWARTZ	14
QUESTIONS, QUESTIONS, QUESTIONS. MARK L CROSBY	16
A PAGE FROM THE STACK: LIBRARIANS CORNER. DAVID MORGANSTEIN	17
UNDELETE THAT FILE ANDY O'BRIEN	18
DISKETTE I/O PROBLEMS. ROBERT H BECKLEY	21
BLAISE AWAY - SPRING PLANTING: SEEDS FOR A TEXT FORMATTER. DR. WO	22
A REVIEW OF LIBRARY DISK VOLUME 32 - GAMES 9 DAVID C STERN	34
CLASSIFIEDS, ADVERTISING RATES	35

ComputerLand®

We know small computers.



 **apple® computer**
Sales and Service

ComputerLand/Tysons Corner
8411 Old Courthouse Road at Rt. 123
893-0424

OFFICERS & STAFF EDITORIAL

President	-Bernard Urban	(301) 229-3458
Vice President	-Rich Wasserstrom	(703) 893-9147
Treasurer	-Bob Peck	(301) 468-2305
Secretary	-Dana Schwartz	(301) 725-6281
Members-at-Large	-Mark Crosby	(202) 488-1980
	-Sandy Greenfarb	(301) 674-5982
	-Hersch Pilloff	(301) 292-3100
Immed. Past Pres.	-John L. Moon	ABBS Sysop
Editor	-Bernard Urban	(above)
Associate Editor	-Rich Wasserstrom	(above)
	-Genevie Urban	(301) 229-3458
Librarians:		
Disk Mail Order	-David Morganstein	(301) 972-4263
Disk Pick-up	-Bill Bowie	(301) 924-3455
Group Purchases	-Howard Lefkowitz	(301) 649-3373
Membership	-Tom Jones	(301) 460-8773
Volunteer Coord.	-Boris Levine	(301) 229-5730
SIG Chairmen:		
ASMSIG	-Jim Rose	(301) 730-2230
EDSIG	-Chuck Philipp	(301) 924-2354
NEWSIG	-Paul Hoffman	(301) 831-7433
Pascal (PIG)	-Tom Woteki	(202) 547-0984
SIGAMES	-Al Gass	(703) 371-3560
SIG/DISABLED	-Curt Robbins	ABBS WAP428
	8805 Barnsley Court Laurel, MD 20811	

Washington Apple Pi
P. O. Box 34511
Washington, D.C. 20034
(301) 621-2719

ABBS (301) 983-9317

Apple user groups may reprint without prior permission any portion of the contents herein, provided proper author, title and publication credits are given.

Membership dues for Washington Apple Pi are \$18.00 per year, beginning in the month joined. If you would like to join, please call the club phone and leave your name and address, or write to the PO Box above. A membership application will be mailed to you.

EVENT QUEUE

Washington Apple Pi meets on the 4th Saturday of each month at 9:30 AM, at George Washington University, usually in Building C, on G Street at 23rd Street, NW. (To be sure of the exact location call the club phone or ABBS during the week of the meeting.)

However, due to the Memorial Day Weekend the May meeting will be on the 5th Saturday, May 30. The June meeting will be on the regularly scheduled 4th Saturday, June 27.

The Executive Board meets on the 2nd Wednesday evening of each month. All members are welcome to attend. Details will be on the club phone and ABBS, or call the President at 229-3458.

NOVAPPLE meets on the 2nd Saturday of the month at 1:00 PM at Kings Park Library on Burke Lake Road in Fairfax County; and on the 4th Thursday of the month at 7:30 PM at Computerland of Tysons Corner.

Thanks... This is my last editorial as President of Washington Apple Pi. You're a good group. Some thoughts about where we've been and where we may go. We have grown to the point where we have lost some of the spontaneity of the monthly meetings. That's inevitable, but can be remedied. During 1979, our first year, information on the APPLE, any information, was in short supply and hard to come by. So we banded together as did others across the country and helped each other out. Now there is almost information overload and we need to sort the dang stuff out. The need is shifting from how to put the APPLE II through its paces to what to use it for. I think that's a good sign - the computer as a means to an end rather than an end unto itself.

Unfinished business - sad to say, I leave for the next President some tasks I had hoped would be undertaken or completed, e.g. tax-exempt status, better documentation of the software, an updated "Who We Are", information kits for prospective and new members, Best of WAP, courses, courses, courses, hard-copy libraries, etc. But we've done remarkably well for an all volunteer group.

Some suggested new business...

- o How about Washington Apple Pi Slices, e.g. WAP-Gaithersburg, WAP-D.C., WAP-Annapolis, etc. so that members who can't or won't come to the Saturday meetings can reap benefits of mid-month meetings in their own "backyard"?

- o How about a WAP office somewhere convenient to serve as pickup (or dropoff) point for software disks, ApNotes, newsletters, the club ABBS, the Hotline, and where members can put in an hour or two of volunteer service?

- o Have we grown so large (about 650 members now) that in order to handle the day-to-day business we need to augment the volunteer service with some paid administrative services?

UPDATE NOTES ON THE EPSON MX-80

by Peter Rosden

During the week of May 11, I received a call from Epson America, in response to a letter I had written inquiring about using the Epson MX-80 with Super Text II, specifically with regard to underlining and superscripting. Epson says that as of June 1, 1981 they will have a PROM for the MX-80 that will allow backspacing for underlining. It will also include other capabilities such as hi-res graphics. They also note that most word processors now include the MX-80. The question of superscripting has not yet been resolved.

MINUTES

EXECUTIVE BOARD MEETING

The Washington Apple Pi Executive Board meeting of April 8, 1981 was called to order at 7:20 PM at the home of the President, with 14 persons in attendance.

Ken Foor, CPA, advised the Board on the progress of our filing for tax-exempt status. The Board revised its plans for the 1981 elections in order to comply completely with the club Constitution and By-laws, with the intent of introducing an amendment which would alter the timing to correspond with our current meeting and publication schedule. The Membership Chairman announced that the membership directory would be republished in June. The Board reviewed and approved a paid newsletter insert for the April issue.

The SIGAMES group reviewed their plans for a programming contest. The need was expressed for the more experienced members to help the new members, and several suggestions for improvement were made. The Board approved funding for the EDSIG to purchase software, not to exceed \$75. Additionally, plans were begun for a club-sponsored seminar on "Using the APPLE".

The meeting was adjourned at 9:50 PM.

GENERAL MONTHLY MEETING

The Washington Apple Pi meeting of April 25, 1981 was called to order at 9:30 AM by the President, with approximately 180 persons in attendance.

The nominations for club officers as proposed at the March meeting were approved by the membership for the May elections.

Opinions were gathered from those present on the possibility of adding additional Members-at-Large to the Executive Board, and the prospects for future increases in the cost of printing.

The main presentation was given on Three-Dimensional Graphics by Andrew Pickholtz, a local high school student.

The meeting was adjourned to SIG meetings at 11:00 AM.

Dana J. Schwartz, Secretary

NOTICES

PLEASE PAY BY CHECK
***** ** ** *****

Members are reminded to bring their checkbooks to the WAP meetings if they plan to do any money dealings. We do not like to carry around large sums of cash. The Management reserves the right to refuse cash in any amount over \$5.00.

1981 BACK ISSUES
**** ** ** *****

We have tried to supply 1981 back issues

to those who have joined since the beginning of the year and for the calendar year. If you have paid \$18.00 and your expiration date (after your WAP #) is 8112, you should have received all 1981 issues. If we have missed anyone, please call Genevieve Urban at 229-3458, or write to the PO Box.

SIG-NEWS

SIGAMES is the special interest group of computer hobbyists interested in using their APPLES for entertainment. The main meeting of this group is held at a location announced at and following the Washington Apple Pi monthly meeting.

A special adventure festival is being planned for this month's meeting. Theron Fuller has scheduled multiple speakers to address different aspects of this subject. Find out those things you always wanted to know about adventures, but didn't know who to ask.

The program in June will consist entirely of young speakers and their impressions of APPLE games. Any young speaker wanting to participate should contact Steve or David Stern at (301) 881-2543.

PIG, the Pascal Interest Group, meets on the third Thursday of each month at 7:30PM at the Uniformed Services University of the Health Sciences, Bldg. A, Room A2054 (2nd floor), on the campus of the National Naval Medical Center at 4301 Jones Bridge Road, Bethesda, MD.

EDSIG will meet on Saturday, May 30, immediately after the regular meeting of Washington Apple Pi.

NEWSIG will meet just after the regular Washington Apple Pi meeting on May 30. The meeting seems to best help the new members by answering their questions, and telling them what to do to get their system up and running. We also tell them something about WAP, how to order the disks, what's on the disks, etc.

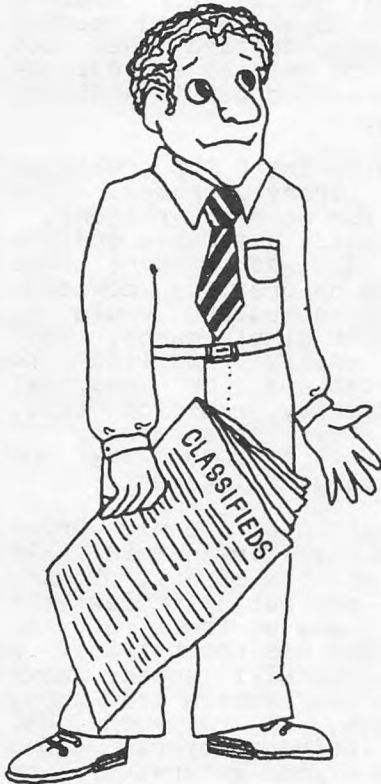
The following members have agreed to answer questions over the phone when someone gets stuck and needs help between meetings:

Bob Chesley 560-0121
Paul Hoffman 831-7433
Sara Lavilla 926-6355
Boris Levine 229-5730
John H. Smith 439-4388
Steve Sondag 281-5392

Greenapples, our SIG for young people, will meet during the regular Washington Apple Pi meeting. After a discussion and planning session, the members will be accompanied by an adult to the APPLE room in the School of Engineering.

FINALLY!

THE USED COMPUTER EXCHANGE IS HERE.



There was a time when buyers and sellers of used computer equipment had nowhere to go for help. Classified ads in local papers were limited and tedious. Access to market trends and current prices were almost non-existent. Sellers often spent long hours with uninformed, uncommitted buyers. A transaction could take weeks to complete.

Now all that has changed.

The Used Computer Exchange offers a unique, nationwide listing service which puts buyers and sellers of used micro-computer equipment together quickly, and you pay only for results!

Our computerized matching service is used by buyers and sellers alike. Just list your specific criteria with us. On the telephone you'll receive the names, phone numbers, and full listings of those who meet your specifications. As others call us, they will be referred to you until you get results.



WHAT ARE SOME TYPICAL MANUFACTURERS?

- APPLE
 - RADIO SHACK
 - COMMODORE
 - IBM 5100
 - NORTH STAR
 - CROMEMCO
 - SHUGART
 - CORVUS
 - LOBO
 - CENTRONICS
 - IDS
 - DIABLO
 - T.I.
- AND MANY OTHERS.

We provide advice about pricing, what to include in an offer, how to handle shipping, and how to protect yourself.

In addition, we offer the *Used Micro-Trends Report* for \$6.75. It includes complete data on key manufacturers and models, price histories, list prices, maintenance costs, details on lowest discount houses and much more.

WHAT CAN I LIST?

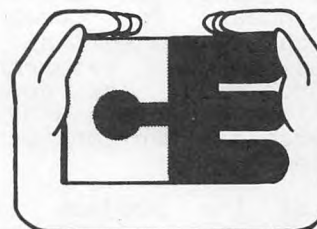
U.C.E. LISTS ANY MICRO-COMPUTER EQUIPMENT WITH A VALUE OF \$300 to \$25,000.

THIS INCLUDES:

- COMPLETE SYSTEMS
- FLOPPY DISK DRIVES
- HARD DISK DRIVES
- PRINTERS
- MONITORS
- TERMINALS

CALL US TOLL-FREE
800-327-9191
(EXTENSION 61)

or (703) 471-0044



**USED
COMPUTER
EXCHANGE**

**THE NATIONWIDE LISTING SERVICE FOR
USED MICRO-COMPUTER EQUIPMENT**

APPLE FOR THE ELDERLY or HOPE FOR CHANGING THE WORLD

by Bernie Benson

We the owners of home computers have a rare opportunity in our lives. Whether we like it or not, we are on the leading edge of a technological revolution that will change the quality of life and social structure of our planet. The change will be no less dramatic than that of the industrial revolution. We are the first generation of humans to use programmable machines in our homes.

The Washington Apple Pi Club and we its members are presently determining the early uses of this new tool. We can set the broadness of the scope through which others will view this new tool. Our early applications will demonstrate whether we see this tool as a new means to pursue old goals or a new tool to tap new areas of creativity never before imagined. We are currently deciding for whom and in what areas of need our efforts will be directed both in the home and in the world. We have the power to determine which human problems will be addressed first and who will reap the benefits of the home computer, at least for the short term. We should not take this responsibility lightly. As a person over 25 still trying to change the world, I find here some very hopeful possibilities.

The APPLE II will be looked on by future generations as one of man's first crude attempts to control the power of information with a micro-electronic device, not unlike the way we look at Edison using a kite, a key, and a mason jar to control electricity, or the Wright brothers using piano wire, bicycle chains and thin wood to control aerodynamic flights.

We will be looked on with envy as the first ones when so many areas of creativity were still untapped. The names most often associated with a specific technology or field of endeavor are those from its early days. Who knows what name might come to mind when a child of the 21st century flips on a microcomputer when he gets up in the morning?

With at least one of these thoughts in mind I offered to help Elaine Eckels with using the APPLE to aid the institutionalized elderly. Dr. Eckels is doing research at Georgetown University and made a request for technical assistance for this project at the February WAP meeting. Her research is addressing the problem of the observed laws of motor skills and mental alertness of many elderly persons after long periods of institutionalized care.

The first goal of the project is to develop some basic programs to measure the mental and physical capacities of the residents at a local nursing home. The

next goal is to develop programs to aid in maintaining and improving these skills. The computer is well suited for dealing with very specific physical and mental limitations. It was decided that the computer would not be used as a replacement for human interaction but in addition to that interaction.

The programs chosen to begin the research came from the WAP library of games. The games were chosen for several reasons. The software was readily available and the price was right. It was thought that elderly persons with no previous knowledge or experience with computers would be least intimidated with simple games. Most games can be easily modified to accommodate limitations in hearing, vision, manual dexterity, reaction time, mental perception, etc. The diversity of the games can attract a broad range of people with widely varying interest.

With the authorization of Bernie Urban (WAP President) and Dave Morganstein (WAP Librarian) and the efforts of Bill Bowie, my wife Paula and I set out to review the approximately 160 games on the first ten games disks. The task was not trivial. A review of these games will appear next month. For now, to new members and those curious about games, I recommend WAP Vol. 3 Games. It includes several types of games and gives a good general flavor of microcomputer games.

We selected 26 games for review by the research group and they selected two for modification and use. The two selected were HANGMAN Vol. 23 and COUNTRY DRIVER Vol. 23. The modifications include enlarging the letters, selecting shorter words, and slowing down the programs.

Dr. Eckels is currently trying to obtain the use of an APPLE computer for this project for the month of July. If anyone can provide any hardware, money or information to assist this effort, please contact Elaine Eckels at Georgetown University (625-7626) or Bernie Benson at home (736-0912). Your generosity will be appreciated and will be productive.

So remember when it's 2 o'clock in the morning and you're looking at a blurry illegal quantity error message and you're frustrated because a new idea didn't turn into a program in just one night, take a minute and think about mason jars, bicycle chains and vacuum tubes and consider some of the similar problems your colleagues had. Consider that, like them, you may be changing the world.

Thanks to Elaine Eckels for giving me a chance to use my knowledge and skills in a new creative worthwhile area.

R-ALPHA SOFTWARE

Box 3332
Crofton, Maryland 21114
Phone (301) 261-3749

the stereo generator

WHAT IS IT? The Stereo Generator is a unique program which allows the Apple user to define, manipulate, and view 3-dimensional objects. Stereoscopic object images created with The Generator may be viewed in FULLY PERCEIVED THREE DIMENSIONS.

HOW DOES IT WORK? The user may define arbitrary 3-dimensional objects comprised of up to 65 points in space which may be connected by up to 170 line segments. The program then generates stereoscopic image pairs on the second high resolution screen, allowing the object to be perceived in three dimensions. Images may be generated in either normal- or mirror- stereo. Separate generation of expanded left and right images is also allowed. Split screen stereo-pairs may be viewed in 3-dimensions on the screen using a mirror. Alternatively, images may be printed and viewed with an inexpensive stereoscope WHICH IS INCLUDED WITH THE PROGRAM. Printed mirror-stereo images may be viewed with a mirror.

CAN THE OBJECT BE CONTROLLED? Yes. Using the game paddles for control, the object may be rotated a full 360 degrees about each of two body axes. (For those familiar with the terminology, the first two Euler angles are under control of the game paddles.) Generation time for a stereoscopic object image varies from one to fifteen seconds, depending upon the complexity of the object. Subroutines allow control of display modes, scale factors, depth perception, object definition, image superposition, orientation sequences, disk control, and printing. Screen images may be saved to disk for printing by any system which can print the second hi-res screen. Object definitions and rotation sequences may also be saved to disk and recalled.

WHO USES IT? Applications for The Stereo Generator are limited only by the imagination. Users include students of geometry and physics, chemists, drafting instructors, graphic artists, and interested hobbyists of all types.

IT'S EXPENSIVE AND HARD TO USE, RIGHT? WRONG! The Stereo Generator costs about half as much as other "3-D" programs which do not in fact produce true 3-dimensional images. The Stereo Generator is a top quality code, professionally written by a physicist. It is user oriented and is accompanied by THOROUGH DOCUMENTATION.

=====

THE STEREO GENERATOR requires a 48K Apple II* or Apple II Plus* and DISK II*. The program will also run on either machine with a Language Card*. Applesoft* language is used. The protected disk can be booted by systems which normally use DOS 3.2. The disk is NOT readable by drives altered for DOS 3.3.

Program, viewer, and documentation: \$27.95 plus \$1.80 for shipping. Maryland residents add 5% tax.

DEALER INQUIRIES ARE INVITED. // User manual available separately, \$5.00 total.

*Trademarks of Apple Computers Inc. // DOS 3.3 will be available June, 1981.

REVIEWS OF THE PASCAL INTEREST GROUP LIBRARY DISKS

by Paul A. Sand

LIBRARY DISK FIG1:

The following is a description of the programs available on this disk. Each program is provided in source code form. The purpose of this minimal documentation is to explain the use of those programs that have no internal directions. I hope you enjoy the use of these routines and that everybody comes up with contributions to future disks.

DSPCHARSET.TEXT - (Bill Wurzel) This program reads the data in SYSTEM.CHARSET, which holds the graphics character set used by the Pascal turtlegraphics routines. It displays any of those characters on the text screen. When started, the program displays character '0' on the screen. To see another character, type '#', <cr>, <the character number>, <cr>. When done, simply type '%', <cr> and the program ends. Note that Bill's program keeps only one block of the two in SYSTEM.CHARSET in memory at once, so when you type in a number greater than 64, the second block is read in. There is no provision to go back to the first block.

DEFCHARS.TEXT - (Bill Wurzel) A logical sequel to the preceding program, DEFCHARS allows you to generate new graphics character sets. When the program is started, SYSTEM.CHARSET is read from drive #4: and copied to a file NEW.CHARSET on drive #5:. (Obviously, those with one drive will have to tinker with this code before using it.) All editing of the character set is done on a 7 x 8 dot matrix. Periods signify off bits, plus signs designate on bits. One creates a new character by moving the cursor through the dot matrix, turning the individual bits on or off as necessary. Single key commands are easy to master: The keys W, E, R, S, F, X, C, V form a cursor pad, moving the cursor in the direction corresponding to the location of the key in the pad. For example, E moves the cursor up, V right and down. The D, B, and M keys are mode-changing keys. D causes bits landed on by the cursor to be turned on, B causes them to be turned off, and the M key causes no change.

Finally, when the character is finished type '*' (if you want to edit more characters or '%' (if you want to exit the program). The program will ask for the number of the character you have just defined, and will save the new definition in NEW.CHARSET. Then, depending on whether you type '*' or '%', the program will allow you to enter another character or exit.

CHECKBOOK.TEXT - (Tom Woteki) Tom's check-book program was presented at the November meeting, and here it is in machine-readable form. It is a program to analyze your checking account cashflow on a monthly basis. The program is (more or

less) self explanatory: running it should pose no problem for you smart guys.

MINIFILER.TEXT - (Tom Woteki) This program duplicates three functions of the UCSD Filer: L)ist Directory, K)runch, and Z)ero Directory. Unlike the Filer, however, you have the source code of this program. By perusing Tom's program, you can learn all sorts of interesting stuff about how directories and files are maintained on your disks. I expect a lot of useful utility programs in the future based on the ideas here.

PEEKPOKE.TEXT - (Michael Hartman) Here are two familiar routines from good old Basic: Peek(addr: integer) returns an integer with the contents of memory location addr. Poke(addr, value: integer) puts value in the location specified by addr. Mike has written these routines in a unit so they can be incorporated into your library.

FILEDUMP.TEXT - (Paul Sand) This is a simple program that allows you to see what is really in those files on your disks. It dumps any file in hex and ASCII, block by block. When started, it asks for a file name. If the file is found, it will display the first half of the first block. Use the arrow keys to display other blocks: back-arrow displays the previous block, forward-arrow displays the next block. Ctrl-C gets you out of the program.

TIMERSTUFF.TEXT - (Paul Sand) These routines are offered to anyone with a Mountain Hardware Apple Clock. They are meant to be used in the program development process to find out where your big, complex program is spending all its time. You can specify timing of up to twenty different portions of code and report at any point the total time recorded by each timer and the number of times each timer was turned on. ("You mean this procedure is called TWENTY THOUSAND TIMES?!"). Procedures included in this unit are:

inittimers - sets everything up,
zeroes timers.
starttimer(i) - starts timer #i
stoptimer(i) - stops timer #i
reporttimers - outputs a table showing the cumulative times, etc.

I don't like these routines much any more - too much stuff to make a very clean interface. But they do work, so if you aren't fussy...

PRINT.TEXT - (Paul Sand) Another simple program, but I use it more than any other. All it does is print out a series of text files on the printer, skipping over page breaks and numbering each page. When started, the program will ask for a file to print. When done, it will ask for another. When finished, just reply with a <cr> to this request.

LIBRARY DISK FIG2:

BIOSUNIT.TEXT
 BIOSDEMO.TEXT
 BIOSDOC.TEXT
 BIOSSTUFF.TEXT - (David Neumann) These files contain the results of David's attempts to figure out how to make Pascal programs do the neat things that are not obvious: inverse characters, horizontal scrolling, etc. Note that the version on this disk works for Apple Pascal version 1.0 only. The same routines for version 1.1 are on library disk FIG3:. David's documentation for these routines is in BIOSDOC.TEXT. BIOSDEMO.TEXT is a nice demonstration of the things his routines allow you to do. BIOSSTUFF.TEXT is the real stuff - 6502 assembly language that does all those things. Finally, BIOSUNIT.TEXT is an example Pascal host unit that could be put in your library.

MENU.TEXT
 MENUDOC.TEXT - (David Neumann) This program will detect the "runable" code files on your disks, display them, and allow you to choose which one to run by typing a single letter command. Very nice. Uses routines in BIOSSTUFF above. MENU.TEXT contains the program, MENUDOC.TEXT contains David's documentation.

DIR.TEXT
 DIRDOC.TEXT - (David Neumann) This program essentially duplicates the extended directory command in the Filer. The program, contained in DIR.TEXT, is a very clear explanation of how one can access the directory from within a program. DIRDOC.TEXT is David's documentation.

CRECHARSET.TEXT
 CRECHARDOC.TEXT - (David Neumann) This program, like DEFCHARS above, allows one to alter the graphics character set. See which one you like better, sports fans. CRECHARSET.TEXT is the program, CRECHARDOC.TEXT is the documentation.

DOODLER.TEXT
 SERENDIP.TEXT
 CUBE.TEXT - (Bill Schultheis) These are three excellent graphics programs. DOODLER is a fast and pretty random drawing routine. SERENDIP is a dot-pattern generator, and CUBE displays a semi-animated cube undergoing rotations, with hidden lines.

(Librarian's note: SERENDIP is based on an algorithm given in the August 1977 BYTE, in which the first advertisements for a computer called the "APPLE II" appeared. A 48K machine "using the new 16K RAM chips" went for the tidy sum of \$2638.00. Ah, the good old days.)

INTDAT.TEXT - () This file is a collection of procedures to convert strings to integers, data strings to integers, and integers back to data strings. This allows one to input data as a string, using inter-line editing, and then convert to numbers. Good idea!

The following is a brief description of each of the files found on this disk. The contributions from the members have been excellent, and I'm sure the users will find a lot of interesting stuff on this disk. I was unable to identify the contributors of some of the programs on this disk. To these anonymoose: Please let me know who you are, so I can give you proper credit in future editions of this disk. I apologize for my sloppy record-keeping and failing memory, and hope you guys don't have easily offended egos.

CRYPTODOC.TEXT
 CRYPTO.TEXT - (Bill Schulteis) The CRYPTO program does cryptanalysis on a level suitable for solving the Sunday Post's magazine puzzle or decoding any KGB transmissions that you happen to pick up on your shortwave. Just kidding. Read CRYPTODOC.TEXT before you try to compile or run CRYPTO - it requires that David Newmann's BIOSSTUFF (from the FIG1: disk, of course. Where have you been?) be installed in your library.

FILER.LIB
 MINIFILER.TEXT - (Tom Woteki) This is Tom's Minifiler described to us at the January meeting. Briefly, it allows filer-like commands to be used from inside your own programs. Tom has made a real achievement here, not only in sheer programming, but also in his exceptionally clean implementation. Note that MINIFILER.TEXT is a procedure, not a complete program. FILER.LIB is a library file that can be used from another program.

PAGEDUMPER.CODE - (Tom Woteki) Dumps out successive pages of your memory in ASCII. you can move both forward and backward in memory, discovering who knows what.

IOUNITA.TEXT
 IOUNITB.TEXT - (Anonymous) These two files contain routines written as intrinsic units that allow your program to do terminal-independent screen I/O by reading terminal data from SYSTEM.MISCINFO. Most, if not all, routines are from APPLE3:DISKIO.TEXT, but this is the right idea - terminal dependencies should not be written into your own programs, but put into the library.

LIFE.TURTL.TEXT
 LORES.USER.TEXT
 LORES.UNIT.TEXT
 LIFE.LORES.TEXT
 LIFE.INFO.TEXT
 LORES.INFO.TEXT - (Michael Hartman) A series of programs and units that implement Conway's Life, both in turtle-graphics (LIFE.TURTL.TEXT) and Lo-res graphics (the others). LIFE.TURTL.TEXT needs to have PEEKPOKE, given in FIG1:, inserted in your system library. LORES.UNIT.TEXT is the Pascal host for the 6502 Assembler code in LORES.USER.TEXT, which is documented in LORES.INFO.TEXT. LIFE.LORES.TEXT is Conway's Life implemented in Lo-res, which uses Mike's lo-res unit. It is documented in LIFE.INFO.TEXT. I sure hope I got all that straight, Mike.

DIS1.TEXT - (William Wurzel) This is Bill's P-Code disassembler, which he described at the December meeting. Obviously, this can be of great use in studies of how the compiler works and how you can write code that takes less space and runs faster. Use is straightforward - good luck!

PLOTPOURRI.TEXT - (Anonymous) A nifty program to do 3-D contour plotting, you might compare it to the Basic version available from the club library. It has a provision to label the plot with the equation of the function plotted, complete with super- and sub-scripts! As an added attraction, it also has a routine to print the whole thing on a Paper Tiger (440 or 445). Fantastic!

RNDSPIRO.TEXT - (Anonymous) A graphics program that draws spirograph-type displays. Very pretty! (Use a color TV.)

LORES.TEXT

PLOT.TEXT

ANDROMEDA.TEXT

DANMAC.TEXT - (David Neumann) What do you know? Another lo-res plotting package! LORES.TEXT is the Pascal host for the 6502 code found in PLOT.TEXT and DANMAC.TEXT. Once you have that together, ANDROMEDA.TEXT is a program that uses the unit.

LIBRARY DISK PIG3:

We again have a disk of useful programs, submitted by our talented membership or filched from other folk.

BIOSDOC.TEXT

BIOSDEMO.TEXT

BIOSTUFF.TEXT

BIOSUNIT.TEXT - (David Neumann, Bill Schultheis) These programs also appear on PIG1, as contributed by David Neumann. Bill Schultheis has modified the assembly language in BIOSSTUFF.TEXT to reflect the changes made by Apple in Pascal version 1.1. I've duplicated the unchanged files here as well. BIOSDOC.TEXT is David's original documentation. BIOSDEMO.TEXT is his demonstration of the various neat things his routines allow you to do: horizontal scrolling of the text page, inverse video characters, text window definition, and other wonders you thought you could never do in Pascal. BIOSUNIT.TEXT is the Pascal unit suitable for inclusion in the system library.

UNASM.INFO.TEXT

XUNASM.CODE - (Bill Schultheis) This is a 6502 disassembler program that will display contents of memory as 6502 instructions, if possible. Supplied in code form only - check it out. UNASM.INFO.TEXT is the file containing Bill's documentation.

X.TRACE.TEXT

X.SPY.TEXT - (Bill Schultheis) These files will show you how to figure out just where your program is taking you in memory. X.SPY.TEXT is the Pascal demo program that turns on tracing for a single Pascal writeln statement. X.TRACE.TEXT is 6502 Assembler code that contains routines needed by SPY. Bill almost makes me wish I knew Assembly language. Almost.

SETUPMX80.TEXT - (Burt Chambers) As one of the proud owners of an Epson MX-80 printer, I can guarantee a warm spot in my heart for anyone who writes software for it. This program is essentially a routine to output those special characters to the MX-80 to allow it to do compressed print and/or overstriking (for darker print). Recommended to all you MX-80 owners. It also poses a challenge to those of you who have other printers: can you provide us with equivalent programs for your own machine, or (better yet) can you write a more general program that would allow a user to define his own printer commands?

BURTS-DUMP.TEXT - (Burt Chambers) An improved version of Tom Woteki's memory dump program (provided on PIG2). Check it out - it may show you where your variables have vanished to.

EIGHTQ.TEXT - (Paul Sand and Somebody Else) This program solves the famous Eight Queens Problem - How can you place eight queens on a chessboard so that no two queens attack each other? I found this program on a San Francisco Apple Core Disk; the documentation states that it came from Washington Apple Pi! It wouldn't quite run successfully as furnished, so I made some pretty extensive changes to it. The pretty recursive algorithm used is taken from Algorithms + Data Structures = Programs by Niklaus Wirth. The exceptionally nice graphics and sound generation were in the original version, however, and I hope whoever it is that wrote it will identify himself (or herself).

DATE.TEX - (Jeff Sue, SF Apple Core) A nice routine to extract the "Last Booted" date from a disk.

MASTERCA.TEXT - (Steve Lloyd, SF Apple Core) - This program creates a "master catalog" file from many disk directories. It could use some fixing up, guys. Although our brother from California has the right idea, we can do better. Soon?

TIGER.UNIT.CODE

TIGER.TEXT

SCRNBYT.TEXT - (Sue, Sue and Gustafson, SF Apple Core) These routines allow a hires screen dump to the venerable Paper Tiger 440/445 printer. TIGER.UNIT.CODE is a unit suitable for inclusion in the system library. TIGER.TEXT is the Pascal source of the unit. SCRNBYT.TEXT is an Assembly language interface to read the hires screen.

PRINTSET.TEXT - (Paul Norris, SF Apple Core) A program to throw the software switches on the Centronics 737 printer. See comments above.

MASTERMI.TEXT - (Paul Norris, SF Apple Core) This program plays Mastermind. I think it cheats.

PILOT.TEXT (George Golden, SF Apple Core) This program is from the July 1980 Byte. It will take a Pilot program and translate it into a Pascal program.

TWOD.TEXT - (David Cheng, SF Apple Core) A Pascal program formatter. It converts

Pascal reserved words and single letter variables into upper case, and some other stuff. See the program for more information.


PRETTYPAS.TEXT - (David Cheng, SF Apple Core) A Pascal program formatter. It converts Pascal reserved words and single letter variables into upper case, and some other stuff. See the program for more information.

BLIZZARD.CODE - (Bob Doran, SF Apple Core) No source file, but one of the prettiest graphics programs I've seen. Will be pleasant to look at next July.

GROCERY.TEXT
MASTER.TEXT - (Jeff Sue, SF Apple Core) Have you ever wanted to use your computer in your everyday life? If so, why? Never mind, you could do worse than to look at this program. Impress the checkout girl at the Safeway with your computer-generated grocery list! A gem of a program as well - techniques used here are very nice.

That's it! Thanks to all contributors, especially the San Francisco Apple Core. And remember...

>>> WALLOW AWAY <<<

PIG Librarian
(Oink!) 

FILE CABINET ENHANCEMENTS by Donald E. Kahler

After the informative talks on the subject of data bases at the March 28 WAP meeting, I had a need to review and modify (for the n'th time) my own version of FILE CABINET. This time I added some of the niceties of the ELECTRONIC FILE CABINET, and exterminated a couple more previously undetected bugs.

In the process, I noticed that the ELECTRONIC FILE CABINET is a relatively universal version, but somewhat limited in the REPORT department. The REPORT section of one of my FILE CABINET versions (I call it FILE CABINET V), included functions and enhancements which, judging by the interest in data bases, might be of value to others in the club.

Briefly, this version:

1. Right-justifies totalled columns
2. Provides for horizontal totals
3. Includes print-using routines
4. Left-justifies non-totalled columns
5. Provides for a UNIT COST column

Item 5 probably requires a little explanation and instruction, since it was devised for a specific purpose of my own, but can be adapted to many applications where you want an average, derived from the previous two columns.

To utilize this UNIT COST or AVERAGE function, you should enter a heading for it when you initiate your data base headings. It must be the last heading established. I didn't make it automatic, like the horizontal total heading, because I found it useful to be able to customize it to the application. When you are entering your data, you just skip over this heading with a 'RETURN'.

Next, when you create your report format, you arrange your UNIT or AVERAGE column on the far right, as indicated above. The 'dividend' column, such as a TOTAL COST column, must be immediately to the left of the UNIT COST column, and the 'divisor' column (total units) must be immediately to the left of the TOTAL COST column.

Incidentally, since I have a serial card for my printer, I have to use the 'SPC' function instead of 'POKE 36' for the tabbing. Therefore, it should work with either parallel or serial systems.

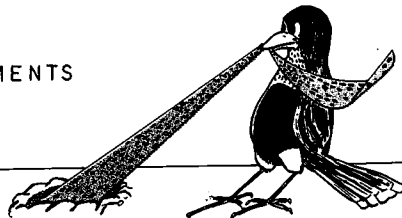
One word of caution, the print-using routines have a width-of-field of 10, so the tabs of your totalled columns must be at least that wide.

In addition to the entire REPORT routine, I am listing below the input subroutine 'borrowed' from the ELECTRONIC FILE CABINET, and the print-using routines.

ROBINS OFFERS A FULL LINE OF SUPPLIES
FOR YOUR MICROCOMPUTER AT VERY
REASONABLE PRICES

And convenient hours:
Tuesday-Friday: 9-5
Monday: 9-7
Saturday: 10-2

ACOUSTICAL ENCLOSURES
ANTI-STATIC MATS
CASSETTES
COMPUTER RIBBONS
COPIER ROLLS
DATA TERMINALS
DISK CARTRIDGES
DISK PACKS
FACSIMILE PAPER
FLOPPY DISKS
FORMS TRACTORS
MAGNETIC CARDS
MEDIA STORAGE
MODEMS
NEC THIMBLES
PERFORATOR TAPE
PINFEED CARDS
PINFEED PAPER
PINFEED LABELS
PRINT WHEELS
RECYCLED RIBBONS
ELECTRIC TYPE ELEMENTS
SURGE SUPPRESSORS
TELETYPE ROLLS
THERMAL PAPER
TYPEWRITER RIBBONS
WORK STATIONS
W-P RIBBONS



Roberts Information Services, Inc.
ROBINS 8306 Hilltop Rd. Fairfax, VA
22031 (in Merrifield) (703)560-5900

```

2940 REM ***REPORT***
2950 T9 = 0
2960 HOME :E = 0
2970 FOR I = 0 TO 3 * NH + 3:K(I) = 0: NEXT I
2980 FOR I = 0 TO NH:AC(I) = 0: NEXT I:HC = 0:GT = 0
2985 PAGE = 0
2990 ON E GOTO 3150
3000 GOTO 3940
3010 IF NH < 19 THEN PRINT "===== "; POKE 34,NH + 5
3014 PRINT : INPUT "HOW MANY HEADERS? ";RH$:RH = VAL (RH$): IF RH < 1 OR RH > NH + 1 THEN 3014
3020 IF E = 0 THEN RN$(NN) = "PRESENT"
3030 FOR I = 1 TO RH * 3 STEP 3
3040 PRINT "ENTER # OF HEADER YOU WANT IN": PRINT "POSITION #"(I + 2) / 3 " "; INPUT "";K$:K(I) =
VAL (K$)
3050 IF K(I) < 0 OR K(I) > NH THEN 3040
3060 PRINT "ENTER TAB FOR "H$(K(I))" "; INVERSE : PRINT ">10 SPCS FOR $"; NORMAL : INPUT "";K$:K
(I + 1) = VAL (K$)
3062 IF I = 1 THEN 3070
3064 IF K(I + 1) = < K(I - 2) THEN 3060
3070 IF K(I + 1) < 0 OR K(I + 1) > 255 THEN 3060
3080 PRINT "TOTAL ON "H$(K(I))"; GOSUB 5500
3090 IF L$ = "Y" THEN K(I + 2) = 1:K(0) = 1
3100 NEXT I
3105 POKE 34,0
3110 IF K(0) < > 1 THEN 3150
3115 PRINT "DO YOU WANT A UNIT COST? "; GOSUB 5600: IF L$ = "" OR L$ = "Y" THEN K(0) = 4: GOTO 31
50
3116 IF L$ < > "N" THEN 3115
3120 INPUT "ENTER TAB FOR HORIZONTAL TOTAL, 'RETURN' FOR NO TOTAL- ";A$
3130 IF LEN (A$) = 0 THEN K(0) = 0:T9 = 1: GOTO 3150
3140 K(I + 1) = VAL (A$): IF K(I + 1) < 0 OR K(I + 1) > 131 THEN PRINT "": VTAB PEEK (37) - 1:
OTO 3120
3150 PRINT "SELECT RECORDS BY WHICH HEADER # "
3160 INVERSE : PRINT "'RETURN' FOR ALL RECORDS": NORMAL :VT = PEEK (37): VTAB VT - 1: PRINT "SELE
CT RECORDS BY WHICH HEADER # "; INPUT "";S$:S = VAL (S$)
3170 IF LEN (S$) = 0 THEN Q$ = "": GOTO 3230
3180 PRINT : PRINT "'AND' 2ND HEADER "; GOSUB 5500: IF L$ < > "Y" THEN X$ = "": GOTO 3200
3190 PRINT : INPUT "ENTER # OF 'AND' HEADER ";X$:X = VAL (X$)
3200 VTAB VT
3210 PRINT : PRINT "SELECT RECORDS FOR "H$(S)"= "; INPUT "";Q$: PRINT
3220 IF L$ = "Y" THEN CALL - 868: PRINT "AND "H$(X)"= "; INPUT "";X$
3230 REM
3246 FOR I = 1 TO RH
3248 IF K(I * 3) = 1 THEN T9 = 1
3249 NEXT I
3250 ON PF GOSUB 5230,5250,5280: GOSUB 3610:Z$ = N$(R(1),S):RR = 0
3260 FOR J = 1 TO NR:Y = R(J)
3270 N$(Y,0) = STR$ (J)
3280 IF Q$ = "" THEN 3320
3290 IF LEFT$ (N$(Y,S), LEN (Q$)) < > Q$ THEN 3330
3300 IF X$ = "" THEN 3320
3310 IF LEFT$ (N$(Y,X), LEN (X$)) < > X$ THEN 3330
3320 GOSUB 3432:Z$ = N$(Y,S)
3330 IF PF < 1 THEN IF L > 18 THEN GOSUB 2060: GOSUB 3610
3331 IF PF > 0 AND J = 1 THEN L = 4
3332 IF J < NR AND L > 55 THEN GOSUB 3610
3340 IF L = 0 THEN GOSUB 3610
3350 NEXT J
3354 IF K(0) = 4 THEN 3360

```



```

3355 GT = GT + HC: GOSUB 3540
3360 IF SW = 1 THEN SW = 0: GOTO 3370: REM TO ELIMINATE DUAL TOTALS
3365 ON T9 GOSUB 3540
3370 GOSUB 5310
3380 ON E GOTO 3410
3390 PRINT : PRINT "DO YOU WANT TO SAVE THE FORMAT": PRINT "FOR THIS REPORT TO DISK? "; GOSUB 55
00
3400 IF L$ = "Y" THEN E = 1: GOSUB 3720
3410 PRINT : PRINT "MORE "RN$(NN)" REPORTS? "; GOSUB 5600
3420 IF L$ < > "N" THEN SW = 0: GOSUB 3880: E = 1: GOTO 2980
3430 GOTO 4810
3432 PK = 0: IF K(0) < > 0 AND K(0) < > 4 THEN IF N$(Y,S) = Z$ THEN PRINT :L = L + 1: IF PF < 1
AND L > 18 THEN GOSUB 2060: GOSUB 3610: GOTO 3437
3435 IF K(0) = 4 OR VAL (PU$) = 0 THEN 3437
3436 IF K(0) < > 0 THEN IF N$(Y,S) < > Z$ THEN PK = K(3 * I - 1) + LEN (H$(K(3 * I - 2))) + 1:
PRINT SPC( PK - SY - LEN (PU$));PU$:GT = GT + HC:HC = 0:RR = 0:L = L + 1: IF PF < 1 AND L > 18 T
HEN GOSUB 2060: GOSUB 3610
3437 IF RR = 0 THEN IF K(0) = 3 THEN PRINT
3438 REM PK=1ST LETTER OF HEADER OR 1ST DIGIT OF AMOUNT - SY=LAST LETTER OF HEADER
3439 PK = 0:SY = 0:SW = 0
3440 FOR I = 1 TO RH
3442 IF K(0) = 4 AND I = RH - 2 THEN UN$ = N$(Y,K(3 * I - 2)):UN = VAL (UN$)
3443 IF K(0) = 4 AND I = RH - 1 THEN UT$ = N$(Y,K(3 * I - 2)):UT = VAL (UT$):QU = UT / UN: GOSUB
6000:QU = UT = UN = 0
3444 IF I = RH AND K(0) = 4 THEN GOTO 3465
3445 IF K(3 * I) = 1 THEN GOSUB 3510
3446 IF K(0) = 2 THEN IF RR < > 0 THEN 3470
3447 IF K(3 * I) = 1 THEN GOTO 3450
3449 IF K(0) = 0 THEN PK = K(3 * I - 1): PRINT SPC( PK - SY);N$(Y,K(3 * I - 2)); GOTO 3460
3450 PK = K(3 * I - 1) + ( LEN (H$(K(3 * I - 2))) - LEN (N$(Y,K(3 * I - 2)))): PRINT SPC( PK - SY
);N$(Y,K(3 * I - 2));
3455 SY = PK + LEN (N$(Y,K(3 * I - 2))): GOTO 3470: REM END OF PREVIOUS PRINT, RIGHT JUSTI
FIED
3460 SY = (PK + LEN (N$(Y,K(3 * I - 2)))): GOTO 3470: REM END OF PREVIOUS PRINT, LEFT JUSTIFIED
3465 PK = K(3 * I - 1) + ( LEN (H$(K(3 * I - 2))) - LEN (QU$)): PRINT SPC( PK - SY);QU$:SY = PK
+ LEN (QU$)
3470 NEXT I
3471 IF K(0) < > 0 AND K(0) < > 4 THEN PK = K(3 * I - 1) + LEN (H$(K(3 * I - 2))) + 1: GOSUB 7000:
PRINT SPC( PK - SY - ( LEN (PU$))):PU$:PU$ = "":GT = GT + HC:HC = 0
3475 IF K(0) = 2 THEN SY = SY - 2
3480 RR = 1
3485 IF K(0) = 2 GOTO 3500
3490 IF K(0) < > 1 THEN L = L + 1: PRINT
3500 RETURN
3510 N = 3 * I - 2
3520 V = VAL (N$(Y,K(N))):AC(I) = AC(I) + V:HC = HC + V
3530 RETURN
3540 FOR I = 1 TO 39 + ((PF > 1) * 39): PRINT "-": NEXT I: PRINT
3542 IF AC(1) = 0 THEN PRINT "TOTALS - ";SY = 9
3545 FOR I = 1 TO RH
3547 AC$(I) = STR$(AC(I))
3549 IF K(3 * I) = 1 AND AC(1) < > 0 THEN SY = 0
3550 IF K(0) < > 4 THEN 3560
3552 IF K(0) = 4 AND I = RH - 2 THEN UN$ = AC$(I):UN = VAL (UN$)
3553 IF K(0) = 4 AND I = RH - 1 THEN UT$ = AC$(I):UT = VAL (UT$):QU = UT / UN: GOSUB 6000:QU = UT
= UN = 0
3554 IF I = RH THEN GOTO 3576
3560 IF AC(I) = 0 THEN 3580
3562 SW = 1
3564 IF K(0) = 4 AND I = RH - 1 THEN NT = AC(I): GOSUB 8000:SY = PK + LEN (PT$): GOTO 3580

```

```

3565 IF I = RH THEN NT = AC(I): GOSUB 8000:SY = PK + LEN (PT$): GOTO 3580
3570 PK = K(3 * I - 1) + LEN (H$(K(3 * I - 2))) - LEN (AC$(I)): PRINT SPC( PK - SY);AC$(I);
3575 SY = PK + LEN (AC$(I)): GOTO 3580: REM      END OF PREVIOUS PRINT
3576 PK = K(3 * I - 1) + ( LEN (H$(K(3 * I - 2))) - LEN (QU$)): PRINT SPC( PK - SY);QU$;SY = PK
+ LEN (QU$):QU$ = ""
3580 NEXT I
3585 IF K(0) = 2 THEN SY = SY - 2
3586 IF K(0) = 0 THEN 3600
3590 GT$ = STR$ (GT): IF GT < > 0 THEN PK = K(3 * I - 1) + LEN (H$(K(3 * I - 2))) + 1: GOSUB 910
0: PRINT SPC( PK - SY - LEN (PU$));PU$;
3600 PRINT : RETURN
3610 HOME :PK = 0:SY = 0
3615 IF Q$ = "" THEN PRINT RN$(NN)" REPORT (ALL RECORDS)"; GOTO 3650
3620 PRINT RN$(NN)" REPORT FOR "H$(S)": "Q$;
3630 IF X$ = "" THEN 3650
3635 PRINT : HTAB 10
3640 PRINT " AND "H$(X)": "X$
3650 PRINT :PAGE = PAGE + 1
3652 PRINT "PAGE ";PAGE
3655 FOR II = 1 TO 39 + ((PF > 1) * 39): PRINT "-";: NEXT II: PRINT
3660 FOR I = 1 TO RH
3670 PK = K(3 * I - 1): PRINT SPC( PK - SY);H$(K(3 * I - 2));
3672 SY = PK + LEN (H$(K(3 * I - 2))): REM      END OF PREVIOUS PRINT
3680 NEXT I
3682 IF K(0) = 4 THEN 3690
3687 IF K(0) < > 0 THEN IF X < > K(3 * I - 1) THEN PK = K(3 * I - 1): PRINT SPC( PK - SY);"TOT
AL";
3690 PRINT
3695 FOR II = 1 TO 39 + ((PF > 1) * 39): PRINT "-";: NEXT II: PRINT
3697 L = 6
3698 IF PF > 0 THEN L = 1
3710 RETURN

5500 PRINT "(Y/";: INVERSE : PRINT "N";: NORMAL : INPUT " ) ";L$: RETURN
5600 PRINT "( ";: INVERSE : PRINT "Y";: NORMAL : INPUT "/N) ";L$: RETURN
6000 PL = 10: REM FIELD WIDTH OF FORMAT
6010 QU$ = RIGHT$ (" " + STR$ ( INT (QU + .0005)) + "." + RIGHT$ ( STR$ ( INT ((QU + 1000) * 1
000 + .5)),3),PL): IF VAL (QU$) < .0001 THEN QU$ = " "
6020 RETURN
7000 PL = 10
7010 PU$ = RIGHT$ (" " + STR$ ( INT (HC + .005)) + "." + RIGHT$ ( STR$ ( INT ((HC + 100) * 1
00 + .5)),2),PL): IF VAL (PU$) < .0005 THEN PU$ = " "
7020 RETURN
8000 PL = 10
8010 PT$ = RIGHT$ (" " + STR$ ( INT (NT + .005)) + "." + RIGHT$ ( STR$ ( INT ((NT + 100) * 1
00 + .5)),2),PL): IF VAL (PT$) < .0005 THEN PT$ = " "
8020 PK = K(3 * I - 1) + LEN (H$(K(3 * I - 2))) - LEN (PT$): PRINT SPC( PK - SY);PT$;
8030 RETURN
9100 PL = 10
9110 PU$ = RIGHT$ (" " + STR$ ( INT (GT + .005)) + "." + RIGHT$ ( STR$ ( INT ((GT + 100) * 1
00 + .5)),2),PL): IF VAL (PU$) < .0005 THEN PU$ = " "
9120 RETURN

```

COST REPORT (ALL RECORDS)

PAGE 1

YEAR	NO	CU/FT	THERMS	COST	\$/THERM
1980	01	222	225.3	89.73	0.398
1980	02	201	203.8	85.15	0.418
1980	03	164	166.3	36.13	0.217
1980	04	75	76.1	36.13	0.475
1980	05	52	52.8	26.39	0.500
1980	06	22	22.3	8.86	0.397
1980	07	13	13.2	5.27	0.399
1980	08	20	20.2	7.90	0.391
1980	09	12	12.2	10.19	0.835
1980	10	30	30.5	18.16	0.595
1980	11	94	95.4	49.27	0.516
1980	12	162	164.4	85.10	0.518
TOTALS -		1067	1082.5	458.28	0.423

⊗

HYPER HEAD-ON: a review

by Walton Francis

In recent months Broderbund Software's Apple Galaxian has risen to the top of the Softalk bestseller list, with nary a mention of Broderbund's HEAD-ON. For my money (\$28), both the reviewers and the buying public are missing the best game yet.

Peelings reviewed HEAD-ON in its January-February 1981 issue, giving it a B+ rating by a reviewer who obviously felt no excitement whatsoever in playing the game. So much for tastes--in my family the game was an instant, addictive hit.

HEAD-ON involves circling a track with multiple lanes. Each lane is initially covered with dots; the short-run object is to travel each lane with your "car" and wipe out the dots. If you get all the dots another race starts, and you can keep going indefinitely. There is only one little problem--the computer sends out a car of its own which tries to hit you head-on, prematurely ending the race. The computer's car is malevolent and clever, changing lanes to meet you and speeding up as you near completion of the race. To the best of my knowledge, this is a game, like Space Invaders, which you can never win. You can win a race, keep your car and race again, but sooner or later the computer car will get you. Also like invader-type games, you have a reserve--a total of four cars (rising to five once you reach a certain point level).

As your score rises (you initially get 5 points for each of 140 dots, plus a bonus for a total score of 1,000 for winning one race), the computer gets tougher. It speeds its car up sooner in each race, and if you win two races before your four cars are wiped out, it starts sending out TWO

cars against you in subsequent races. If you get skillful enough to beat two cars, the computer has other nasty surprises in store (I don't want to give it all away--and it takes many dozens or hundreds of games to be able to beat two cars consistently).

Success in this game requires both intellect and manual dexterity, plus nerves of steel. There are subtle tactical approaches which enable even a fumble fingers to beat the fastest kid on the block, but a real pro needs it all.

The program allows you to control your car's direction from the keyboard, with paddles, or with a joystick. A real defect, or a virtue, depending on your point of view and your wallet, is that only a joystick allows you to achieve real proficiency and roll up scores in the 10,000 plus range. In addition to your control over your car's direction, you must also learn to use the paddle or joystick button to control speed--learning when to go slow and when to go fast is vital to success, and only maximum speed will enable you to get those last few dots in each race.

HEAD-ON is a HIRES graphics game, with multiple colors and nice sound effects. It records the score achieved in each game, which allows your family to take turns and see who can get the best score. Its only real defect is that it can be quite frustrating to master the game sufficiently not to be wiped out rapidly--it takes MANY games to achieve any real degree of skill. All in all, a super game which I cannot recommend too highly.

⊗

SINGLE DRIVE CONVERT

DOS 3.2 → 3.3 by Dana J. Schwartz

Everybody talks about converting their disks from DOS 3.2 to 3.3, but nobody ever does anything about it! Well, I can name that tune in 5 passes (or less!).

The Spring 1981 Apple Orchard contained a fine single-drive copy program by Steve Adams written in Integer Basic (SINGLE DISK COPY.3). I took that routine and integrated it with MUFFIN to produce a single-drive 3.2 to 3.3 conversion program which does not require you to switch disks after each file (as MUFFIN alone does). A 48K Apple II has sufficient RAM to transfer 100 sectors on each pass, which will convert a full 3.2 disk in five passes.

To use the program it should be kept on a 3.3 disk along with a copy of MUFFIN from your 3.3 master. The disk which is to receive the files from the original 3.2 disk should have been previously initialized under 3.3. Any files existing on the new disk will be overwritten and lost. After starting SINGLE DRIVE CONVERT you will be prompted as to which disk to insert at each point. All the cautions stated in the original Apple Orchard article are still valid, including the prohibition on changing HIMEM: and write-protection of the copy disks.

Since the original program was adequately explained in the Apple Orchard article, I will only attempt to explain below the modifications which I have introduced.

Line 0 sets LOMEM: to 8460 to reserve space for MUFFIN, which is loaded at line 500. Note that LOMEM: must be set before any variables are defined (line 1).

The comments before line 1000 were moved to the end for speed purposes, and PRINTing in lines 1000-1010, 4090, and 4100 was changed to reflect the new version. (The speed didn't seem to improve, but in my heart I know I was right.)

Lines 1067, 4035, and 4037 change DOS pointers to the appropriate RWTS (Read or Write a Track and Sector) routine for 3.2 or 3.3. This is the basis of the conversion process, allowing DOS 3.3 to read 13-sector (3.2) disks.

Since only 3.2 disks will be input, lines 1, 1080, and 2040-2095 were changed to remove the variable VER. Also, as we don't want to copy the 3.2 DOS onto the 3.3 disk, lines 2010 and 2040 were altered to only transfer tracks 3 through 34, leaving out tracks 0-2.

Lines 2100-2120 were added to convert the 3.2 bitmaps to 3.3 bitmaps in the VTOC. The conversion is completed by a GOSUB in line 4070 to a new subroutine (lines 6000-6110) which rewrites the VTOC and

readjusts the links in the directory. Also, line 4070 allows for a restart for multiple conversions, if desired.

Note that disks without standard DOS directories (track 17) and most "uncopyable" disks don't stand a niffum of a chance of being converted successfully. (Hey, that gives me another idea...)

>LIST

```
0 POKE 74,8460 MOD 256: POKE
75,8460/256: POKE 204, PEEK
(74): POKE 205, PEEK (75): REM
LOMEM:8460
1 A=B=PTR=LOC=RWTS=IBTRK=IBSECT=
IBBUFP=REP=REPS=CMD=TBL=IOB=
TRK=SEC=BYTE1=BYTE2=OLDPTR=
BITMAP=BUFLO=BUFHI=IBVOL=IBCMD=
0
500 PRINT "BLOAD MUFFIN": REM CTRL-D
1000 TEXT : CALL -936: VTAB 4: TAB
11: PRINT "SINGLE DRIVE CONVERT"
: TAB (16): PRINT "3.2 -> 3.3"
: VTAB 7: TAB 11: PRINT "BY DANA
J. SCHWARTZ"
1005 TAB 11: PRINT "WASHINGTON APPLE
PI": VTAB 12: TAB 16: PRINT
"BASED ON": TAB 11: PRINT "SINGL
E DRIVE COPY.3": TAB 13: PRINT
"BY STEVE ADAMS"
1010 VTAB 20: PRINT "INSERT THE DISK(
3.2) YOU WISH TO CONVERTAND GENT
LY TOUCH RETURN TO BEGIN.":
GOSUB 5010
1020 REM
1021 REM *** FIND THE IOB ***
1022 REM
1030 A= PEEK (77): IF A>94 THEN
A=A-256:IOB=(A+33)*256+231
1040 REM
1041 REM *** LOAD CONTROLLING ***
1042 REM *** SUBROUTINE IN ***
1043 REM *** PAGE 0 ***
1044 REM
1050 POKE 0,169: POKE 1,IOB/256+
255*(IOB<0): POKE 2,160: POKE
3,232: POKE 4,32: POKE 5,217
: POKE 6,3: POKE 7,96
1060 REM
1061 REM *** LOAD THE VTOC INTO ***
1062 REM *** MEMORY STARTING ***
1063 REM *** AT $02D0 ***
1064 REM
1065 IBVOL=IOB+4:IBTRK=IOB+5:IBSECT=
IOB+6:IBBUFP=IOB+10:IBCMD=IOB+
13
1067 POKE -17152,76: POKE -17151
,0: POKE -17150,30: REM MUFFIN
RWTS
1070 POKE IBVOL,0: POKE IBTRK,17
: POKE IBSECT,0: POKE IBBUFP-
1,208: POKE IBBUFP,2: POKE
IBCMD,1: CALL RWTS
1080 BITMAP=776
2000 REM
2001 REM *** THE TABLE STARTS ***
2002 REM *** AT "TBL" ***
2003 REM
2010 TBL= PEEK (204)+ PEEK (205)
*256+1:PTR=TBL
2020 VTAB 10: TAB 15: PRINT "I'M THIN
KING"
```



```

2030 REM
2031 REM *** READ "BIT MAPS" IN ***
2032 REM *** THE VTOC AND CON- ***
2033 REM *** VERT TO BINARY ***
2034 REM
2040 FOR TRK=3 TO 34:BYTE1=BITMAP+
TRK*4:BYTE2=BYTE1+1:SEC=12
2050 A= PEEK (BYTE1): IF A#255 THEN
2070:SEC=4
2060 A= PEEK (BYTE2): IF A=248 THEN
2100
2070 B=A/128:A=A-B*128: GOSUB 3020
:B=A/64:A=A-B*64: GOSUB 3020
:B=A/32:A=A-B*32: GOSUB 3020
:B=A/16:A=A-B*16: GOSUB 3020
:B=A/8:A=A-B*8: GOSUB 3020
2080 IF SEC<0 THEN 2100
2090 B=A/4:A=A-B*4: GOSUB 3020:B=
A/2:A=A-B*2: GOSUB 3020:B=A:
GOSUB 3020
2095 GOTO 2060
2100 IF TRK<3 THEN 2120
2110 A= PEEK (BYTE1):B= PEEK (BYTE2)
: POKE BYTE1,A/8+224: POKE
BYTE2,B/8+(A MOD 8)*32: REM CON-
VERT BITMAP
2120 NEXT TRK: GOTO 4010
3000 REM
3001 REM *** IF THE TRACK BIT ***
3002 REM *** MAP INDICATES AN ***
3003 REM *** IN-USE SECTOR, ***
3004 REM *** POKE TRK & SEC ***
3005 REM *** INTO THE TABLE ***
3006 REM *** STARTING AT "TBL" ***
3007 REM
3020 IF B THEN 3030: POKE PTR,TRK:
POKE PTR+1,SEC:PTR=PTR+2
3030 SEC=SEC-1: RETURN
4000 REM
4001 REM **** COPY ****
4002 REM
4010 BUFLO=(PTR) MOD 256:BUFHI=(
PTR)/256: POKE IBBUFP-1,BUFLO:
POKE IBBUFP,BUFHI
4020 REPS= PEEK (203)-BUFHI-( PEEK
(202)<BUFLO):OLDPTR=TBL:TBL=
PTR
4030 FOR CMD=1 TO 2: CALL -936: IF
CMD=1 THEN PRINT "READING":
IF CMD=2 THEN PRINT "WRITING"
: POKE IBCMD,CMD:LOC=BUFHI:
PTR=OLDPTR
4035 POKE -17152,76: POKE -17151
,0: POKE -17150,30: REM MUFFIN
RWTS
4037 IF CMD=1 THEN 4040: POKE -17152
,132: POKE -17151,72: POKE
-17150,133: REM 3.3 RWTS
4040 FOR REP=1 TO REPS: POKE IBTRK,
PEEK (PTR): POKE IBSECT, PEEK
(PTR+1): POKE IBBUFP,LOC
4045 VTAB 3: PRINT "TRACK=": PEEK
(IBTRK): TAB 12: PRINT "SEC="
: PEEK (IBSECT): " "
4050 CALL RWTS
4060 LOC=LOC+1:PTR=PTR+2: IF PTR#
TBL THEN 4080
4070 IF CMD=1 THEN 4090: GOSUB 6010
: CALL -936: PRINT "FINISHED"
: PRINT : INPUT "ANOTHER DISK (1
=Y/0=N)",A: IF A=1 THEN 1000
: END
4080 NEXT REP
4090 FOR A=1 TO 1000: NEXT A: CALL
-936: VTAB 5: PRINT "INSERT THE
": IF CMD=1 THEN PRINT "DUPLICA
TE(3.3)":
4100 IF CMD=2 THEN PRINT "ORIGINAL(3.
2)": PRINT " AND HIT RETURN"
: GOSUB 5010
4110 NEXT CMD:OLDPTR=PTR: GOTO 4030

```

```

5000 REM
5001 REM *** WAIT FOR 'RETURN' ***
5002 REM
5010 POKE -16368,0
5020 IF PEEK (-16384)#141 THEN 5020
: POKE -16368,0: CALL -936:
RETURN
6000 REM
6001 REM *** HANDLE TRK 17 ***
6002 REM
6010 CALL -936: VTAB 3: PRINT "REVISI
NG VTOC & DIR"
6020 POKE 722,15: POKE 723,3: POKE
726,254: POKE 773,16: POKE
844,0
6030 POKE IBTRK,17: POKE IBSECT,
0: POKE IBBUFP-1,208: POKE
IBBUFP,2: CALL RWTS: REM REWRITE
VTOC
6040 FOR A=12 TO 1 STEP -1
6050 POKE IBSECT,A: POKE IBCMD,1
: CALL RWTS: REM READ DIR
6060 POKE 721,17: POKE 722,A+2: POKE
IBSECT,A+3: POKE IBCMD,2: CALL
RWTS: REM SHIFT & REWRITE
6070 NEXT A: FOR A=720 TO 975: POKE
A,0: NEXT A: REM DO LAST 3
6080 POKE IBSECT,1: CALL RWTS
6090 POKE 721,17: POKE 722,1: POKE
IBSECT,2: CALL RWTS
6100 POKE 722,2: POKE IBSECT,3: CALL
RWTS
6110 RETURN
9000 REM *****
9010 REM *
9020 REM * SINGLE DRIVE CONVERT *
9030 REM * 3.2 -> 3.3 *
9040 REM *
9050 REM *****
9060 REM *
9070 REM * BY DANA J. SCHWARTZ *
9080 REM * WASHINGTON APPLE PI *
9090 REM *
9100 REM * BASED ON *
9110 REM * SINGLE DRIVE COPY.3 *
9120 REM * BY STEVE ADAMS *
9130 REM *APPLE ORCHARD SPRING 1981*
9140 REM *
9150 REM *****

```

GROW WITH THE LEADER

Join our team! Be a part of the fastest growing distributor of microcomputer products on the East Coast. Our high volume operation has created personnel needs in several key areas. If you think you would be interested in growing with the leader, we want to hear from you. Send us your resume, and include a letter about your interests.

ATTN: R. BEALLE
MICRO DISTRIBUTORS, INC.
11794 Parklawn Drive
Rockville, MD 20852

WE ARE NOT A RETAIL OUTLET.

PLEASE NO CALLS

Questions, Questions

Questions by Mark L. Crosby

Q. Having but one drive for the last year, I was so-o-o happy to finally get drive #2. However, when I plugged it into its proper connector on the interface card and tried out the 2 drive system, I was not able to get drive 2 to do anything but turn on its motor, whir, and return with an I/O error. Drive 1 booted and worked fine as ever. Switching the drives pointed to connector #2 on the control card as the problem. Do you think I should take the offensive control card to a dealer to fix?

A. There are many possibilities why things are not working as they should. Unfortunately, it is just too easy to blow out chips with static charges to be sure what the problem might be. The dealer diagnostics disk won't help much here either since it relies on a good controller card. Chances are you'll have to take it in for repair anyway, but first try it in a friend's Apple with his/her DOS disk and drives (not yours) just to be sure it is the card.

Q. I own an IDS 440G printer which is interfaced to an Apple II by way of the game I/O connector. Are there any decent, relatively fast hi-res dump programs available designed to handle such a setup?

A. In Vol 1 # 11 (DEC 1979) of WAP is an article and program by Hersch Pilloff to do just that.

Q. Is it possible to break from a running program and modify some of its program lines without (this is the important part) wiping out the values of the current program variables?

A. Since I assume you mean Applesoft programs, you first need to protect an area for your variables by allowing program expansion space. By using the two short EXEC files that follow, you can restore variables that are normally lost when you modify a program. To use these utilities, first write a program. You must set LOMEM: to a higher value to allow extra space for newly added program lines. I do this: type: PRINT PEEK (106) <CR>, add 8 to the result and then POKE 106, RESULT. This gives you 2048 extra bytes of program space. Otherwise you will write over the variable space. Then run the program and print the values of some of your variables. Now type: EXEC SAVE POINTERS <CR>. This saves the necessary pointers to all your variables and strings. You may now make program changes. If you print variables now they will be zero or null. To restore the previously calculated variables type: EXEC RESTORE POINTERS <CR>. Here are the two programs that create the two EXEC files:

```
10 D$ = CHR$(4)
20 PRINT D$"OPEN SAVE POINTERS"
30 PRINT D$"WRITE SAVE POINTERS"
40 PRINT "POKE 0, PEEK (105)"
50 PRINT "POKE 1, PEEK (106)"
60 PRINT "POKE 2, PEEK (107)"
70 PRINT "POKE 3, PEEK (108)"
80 PRINT "POKE 4, PEEK (109)"
90 PRINT "POKE 5, PEEK (110)"
100 PRINT "POKE 6, PEEK (111)"
110 PRINT "POKE 7, PEEK (112)"
120 PRINT "POKE 8, PEEK (115)"
130 PRINT "POKE 9, PEEK (116)"
140 PRINT D$"CLOSE"
```

```
10 D$ = CHR$(4)
20 PRINT D$"OPEN RESTORE POINTERS"
30 PRINT D$"WRITE RESTORE POINTERS"
40 PRINT "POKE 105, PEEK (0)"
50 PRINT "POKE 106, PEEK (1)"
60 PRINT "POKE 107, PEEK (2)"
70 PRINT "POKE 108, PEEK (3)"
80 PRINT "POKE 109, PEEK (4)"
90 PRINT "POKE 110, PEEK (5)"
100 PRINT "POKE 111, PEEK (6)"
110 PRINT "POKE 112, PEEK (7)"
120 PRINT "POKE 115, PEEK (8)"
130 PRINT "POKE 116, PEEK (9)"
140 PRINT D$"CLOSE"
```

Q. I have heard that some software publishers protect their software by interleaving tracks. Are there more than 35 tracks on a diskette?

A. Track seeking is controlled by software, not by hardware. It is therefore possible to position the read/write head anywhere on the disk (within the boundaries). There are several commercial disks which are protected in this manner. Several tracks are written in their proper place and the rest of the tracks are shifted one-half-a-track using special software to control head movement. It is generally effective but there are some copy programs which copy half-tracks too. There is also a company which sells a drive having 70 tracks (using accurate band-positioning), thereby doubling the present capacity of a diskette.

Q. How do I center text on the screen when I don't know the length of a string?

A. Let Applesoft figure it out for you. Assuming your string is equal to or less than 40 characters in length and located in A\$ the formula is: HTAB 21 - INT (LEN (A\$) / 2 + .5): PRINT A\$. This can also be accomplished on your printer where the number of characters per line (i.e., 80, 96, 132, etc.) is n. POKE 36, ((n/2) - INT (LEN (A\$) / 2 + .5)): PRINT A\$. Here is a simple demonstration:

```
10 B$ = "/"
20 A$ = B$
30 FOR I = 1 TO 40
40 HTAB 21 - INT ( LEN (A$) / 2 + .5)
```

50 PRINT A\$
60 A\$ = A\$ + B\$
70 NEXT I

- Q. Where can I get a complete catalog of integrated circuits?
- A. The one I turn to most often is the Newark Electronics catalog (752 pages). They carry a wide variety of parts, tools and test equipment. Newark Electronics, 6232 N. Pulaski Road, Chicago, IL 60646 (312) 286-4700.

There are a number of questions which have been raised that have not been answered. Anyone who would care to respond to these will have their response published in the next available issue. Send answers to our PO Box.

- Q. I recently bought an Apple along with ASCII Express. The problem I have is when I'm using it with another computer terminal attached to my Apple so I can get 80 columns - it disconnects the terminal. Is there a simple way to use both my terminal and ASCII Express?
- Q. What programming manipulations must a person make on his/her Apple II Plus in order to discover the number of free sectors left on a disk? I want to include this as a subroutine within an Applesoft Program.
- Q. How in the world do I get past the Cyclops in Zork? Also, is there any way to get underneath the grating (I have the key)?

⊗

A PAGE FROM THE STACK: LIBRARIAN'S CORNER by Dave Morganstein

New for this month is Disk 37, a utility disk in 3.2. On the disk will be: a corrected version of Amper-interpretor from Nybble, a machine language version of Paul Sand's Prettylisting program converted by Bill Schultheis, and a new Super File Cabinet which contains a machine language sort routine for very fast sorting. The disk can be ordered any time after the May meeting for mail out or pickup at the June meeting.

In a separate article by Paul Sand, you can find a description of the Pascal Interest Group's disks, PIG1: - PIG3:.. Also, a review of Volume 32, written by David Stern, will provide some reaction to our Games 9 disk.

COMMERCIAL SOFTWARE

Operation Apocalypse & Warp Factor (SSI).

Since these are two new games from the people at SSI you might wonder why I want to review them in the same breath. Well, I find that they provide an interesting comparison of what I like and don't like in games, and I thought the comparison useful. Several months ago I raved about Computer Conflict (also from SSI) and indicated that I would like to see a slightly more complicated version. Well, Apocalypse is just about everything I had hoped for in a more challenging package. It provides for solitaire or partner games, it has several different scenarios, it has a more complicated multi-colored display (now including rivers and bridges). Also, many new features have been added, such as artillery both on and off board, paratroop landings, landing craft and distinction between night time and day time moves. There are more types of troops (tanks, infantry, engineers, and artillery) and they can be in different modes with differing consequences. While in transport, they have reduced fighting ability. While in combat, there is reduced mobility. While reorganizing and regaining strength and mobility they suffer reduced defensive ability. To put it mildly, I think it is an excellent simulation.

In the same period, SSI puts Warp Factor on sale. While the game has detailed simulation of space combat it suffers from many shortcomings in comparison with Apocalypse. The game is all in black and white. While the hi-res screen is used to display the Universe under various magnifications, the display is static and mostly black space (yes, I know space is mostly balck space. One point for reality, zero for jazzy graphics!!!). The game moves slowly through overlong pauses between overlays. Your job is to allocate yours ships' energy to a dozen different systems and then plan your attack pattern. Many different ships can be used, each with different defensive and offensive weapons. However, to understand this you must study the flash cards provided and read the wordy descriptions. When battle does come, there is very little by way of sound or graphics. You cannot easily tell why your shots missed, as there is no display other than a slow scrolling hi-res character generator display of text. Not terribly impressive...The game deserves plaudits for detail and accuracy but I did not find it holding my attention.

VU #3. (Progressive Software).

If you use VISICALC you may find this batch of utility programs quite useful. The main function of the programs are to allow interchanging VC files with your own basic programs. Thus, if you have a File Cabinet file or a block of data downloaded from a main-frame you can use VU #3 to convert it for use in VC. Similarly, a VC file can be made available for use in the basic environment. Other uses allow for manipulating rows or columns of VC files and re-entering them into VISICALC.

⊗

UNDELETE THAT FILE

by Andy O'Brien

Occasionally, even the greatest programmer deletes a file that he really didn't mean to delete. If he is lucky, he can whip out his trusty backup diskette and restore the file. If he is not lucky, he can crack his knuckles and begin reentering the file.

Well, great programmers, have no fear, UNDELETE is here. UNDELETE can temporarily restore a deleted file. What I mean by temporarily, is that this utility must be used right after the file is deleted and that it does not restore the VTOC to its original state (before the delete).

Consider the following scenario. I have just deleted an Integer Basic file called MASSIVE, when I meant to delete a file called MISSILE. I quickly do a 'BLOAD UNDELETE' and a 'CALL 768'. When prompted for a filename, I enter 'MASSIVE'. I next do a 'CATALOG' and see that the file has been restored to the directory. But my job is not finished since I know that

UNDELETE does not touch the VTOC, and the sectors that make up the file MASSIVE are currently flagged as being free to DOS. To finish up, I 'LOAD MASSIVE' to get the file into memory, 'DELETE MASSIVE' to remove the directory entry, and finally I 'SAVE MASSIVE' to restore the file.

I wrote this program in order to become more familiar with DOS and with programming the APPLE Assembly Language. And while I admit that it is a bit cumbersome to restore a file in this manner, consider the alternatives. It could be a lifesaver.

Once again, the following five steps are necessary to restore an accidentally deleted file:

1. BLOAD UNDELETE
2. CALL 768
3. LOAD the file
4. DELETE the file
5. SAVE the file

:ASM

```
*****
2      *                UNDELETE.ASM                *
*****
4      *
5      *
6      RECLN          EQU    $23
7      DATRWL         EQU    $00
8      DATRWH         EQU    $96
9      LINLEN         EQU    $10
10     FROML          EQU    LINLEN+$1
11     FROMH          EQU    LINLEN+$2
12     LINBUF         EQU    $200
13     RWTS          EQU    $3D9
14     IOB           EQU    $B7E8
15     EXPVOL        EQU    IOB+$3
16     TRACK         EQU    IOB+$4
17     SECTOR        EQU    IOB+$5
18     RWBUFL        EQU    IOB+$8
19     RWBUFH        EQU    IOB+$9
20     COMM          EQU    IOB+$C
21     PUTMSG        EQU    $A702
22     GETLIN        EQU    $FD6F
23     CROUT         EQU    $FD8E
24     COUT          EQU    $FDED
25     BELL          EQU    $FF3A
26     *
27     *
28     ORG           EQU    $300
29     OBJ           EQU    $7000
30     *
31     *
```



```

*****
33 * Initialize variables
*****
35 *
0300: D8 36 START CLD
0301: A9 0F 37 LDA #$F Setup for first track
0303: 8D ED B7 38 STA SECTOR of directory,
0306: A9 11 39 LDA #$11
0308: 8D EC B7 40 STA TRACK
030B: A9 96 41 LDA #DATRWH Tell DOS where RWTS
030D: 8D F1 B7 42 STA RWBUFH buffer is.
0310: A9 00 43 LDA #DATRWL
0312: 8D F0 B7 44 STA RWBUFL
0315: A9 00 45 LDA #$0 Expected volume number
0317: 8D EB B7 46 STA EXPVOL (@ doesn't try to match).
47 *
48 *
*****
50 * Prompt for filename
51 * to undelete
*****
53 *
031A: 20 8E FD 54 JSR CROUT
031D: A0 00 55 LDY #$0
031F: B9 A0 03 56 PR1 LDA FILMSG,Y
0322: 20 ED FD 57 JSR COUT
0325: C8 58 INY
0326: C0 09 59 CPY #$9
0328: D0 F5 60 BNE PR1
032A: 20 6F FD 61 JSR GETLIN Monitor routine to get a line.
032D: 86 10 62 STX LINLEN X register has the line length.
63 *
64 *
*****
66 * Find the file
*****
68 *
032F: A9 01 69 GETDIR LDA #1 Read the current sector.
0331: 8D F4 B7 70 STA COMM
0334: A9 B7 71 LDA #$B7
0336: A0 E0 72 LDY #$E0
0338: 20 D9 03 73 JSR RWTS
033B: A9 0E 74 LDA #$E Setup for first filename
033D: 85 11 75 STA FROML in the sector.
033F: A9 96 76 LDA #DATRWH
0341: 85 12 77 STA FROMH
78 *
0343: A0 03 79 CHKEND LDY #$3 Is this the last entry in
0345: B1 11 80 LDA (FROML),Y the directory?
0347: C9 00 81 CMP #0
0349: D0 0F 82 BNE CHKDEL No!
034B: 20 8E FD 83 JSR CROUT Yes, tell user that the file
034E: 20 3A FF 84 JSR BELL isn't here.
0351: A2 06 85 LDX #$6
0353: 20 02 A7 86 JSR PUTMSG
0356: 20 8E FD 87 JSR CROUT
0359: 60 88 RTS
89 *
035A: A0 00 90 CHKDEL LDY #$0 Is this entry deleted?
035C: B1 11 91 LDA (FROML),Y
035E: C9 FF 92 CMP #$FF
0360: D0 14 93 BNE OUT No!

```

```

0362: A2 00 94          LDX  #0          Yes!
0364: A0 03 95          LDY  #3          Is the entered filename the
0366: B1 11 96          LDA  (FROML),Y   same as the current
0368: D0 00 02 97      HERE          CMP  LINBUF,X   directory entry?
0368: D0 09 98          BNE  OUT         No!
036D: C8 99          INY
036E: E8 100          INX
036F: E4 10 101         CPX  LINLEN
0371: D0 F3 102         BNE  HERE
0373: 4C 85 03 103      JMP  GOTIT       We got a match!
0376: 18 104          OUT          CLC
0377: A5 11 105          LDA  FROML       Setup for next file name.
0379: 69 23 106          ADC  #RECLN
037B: 85 11 107          STA  FROML
037D: 90 C4 108          BCC  CHKEND
          109      *
037F: CE ED B7 110      NEWSCT          DEC  SECTOR     Setup for next sector.
0382: 4C 2F 03 111      JMP  GETDIR
          112      *
          113      *
          *****
          115      * Undelete it
          *****
          117      *
0385: A0 20 118          GOTIT          LDY  #20         Move bytes around to
0387: B1 11 119          LDA  (FROML),Y undelete file.
0389: A0 00 120          LDY  #0
038B: 91 11 121          STA  (FROML),Y
038D: A9 A0 122          LDA  #A0
038F: A0 20 123          LDY  #20
0391: 91 11 124          STA  (FROML),Y
0393: A9 02 125          LDA  #2
0395: 8D F4 B7 126      STA  COMM       Write out current sector
0398: A9 B7 127          LDA  #B7         with temporarily restored
039A: A0 E8 128          LDY  #E8         directory entry
039C: 20 D9 03 129      JSR  RWTS
039F: 60 130          RTS
          131      *
          132      *
          *****
          134      * Messades
          *****
          136      *
03A0: C6 C9 CC 137      FILMSG          RSC  'FILENAME:'

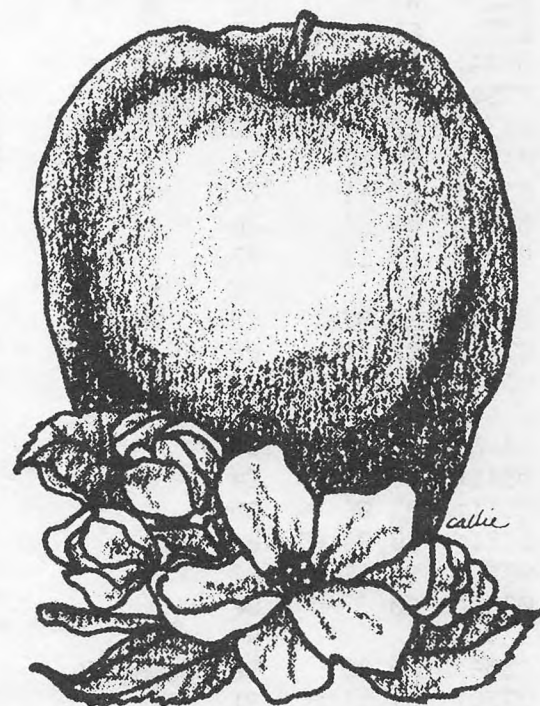
```

--- END ASSEMBLY ---

TOTAL ERRORS: \$00

CODE STARTS AT \$7000
 ENDS AT \$70A8
 RANGE = \$00A9

\$00A9 BYTES OF OBJECT CODE GENERATED.



BLAISE AWAY

by Dr. Wo

SPRING PLANTING:

SEEDS FOR A TEXT FORMATTER

For some time now I've been meaning to write some kind of text formatter. To be sure, there are some good word processors and text formatters, available. There are even a few (good ones, I'm told) written in Pascal: the Moonshadow Text Formatter distributed by Computers Plus Inc.; the program 'prose', available through the USUS library and the Source Apple Users Group (SAUG); and a version of good old NROFF being developed by Ned Rhoades of Washington, D.C. However, if you're like me, running someone else's programs is a little like wearing someone else's boots: your own always feel so much better.

What finally set me down to actually write a formatter program was the program 'list' appearing on the International Apple Core disk 'ATTACH:'. The program is designed to do one thing, write text files to the printer, and has a very useful feature, the ability to "include" files (like the compiler does) using a classic data structure, the stack. No doubt the program was included on the disk specifically to facilitate printing the extensive BIOS documentation stored on ATTACH:.

I tried 'list'. It ran OK but some of the parameters were not set up right for my printer, a Paper Tiger. I started tinkering with the program but found I didn't like the way the code was laid out so I had a hard time modifying it. I decided then and there to write my own program borrowing features from 'list' and NROFF. The result was 'tformat'.

The Program

'tformat' is a simple line oriented text formatter with a fair amount of error checking designed to keep the program from walking away from you. It takes disk-based text files with imbedded formatting commands as input

and sends formatted text to the console, the printer or named disk files. It's fast enough in that it has no trouble keeping my Paper Tiger busy on output-- I don't have to wait while the printer stops and starts.

'tformat' is best used in conjunction with the Pascal text editor and its 'margin' and 'filling' features. Set 'auto' to false, 'filling' to true, set the left, right and paragraph margins to desired values and use the 'margin' command to fill and spruce up paragraphs before submitting files to 'tformat'. Set the pagewidth in 'tformat' (described below) to right-left1.

The Commands

=====

All commands are strings of the form:

tXY or

tXYn or tXY+n or tXY-n or

tXYstring

where 'XY' is a two character mnemonic for the command, 'n' is an integer and 'string' is a string such as a file name. Alphabetic characters may be upper or lower case since the program capitalizes all command strings. Each command must begin on a new line and the first character, including blanks, must be the command character 't', the constant 'commandprefix' in the program.

The following commands are implemented:

pagewidth tPw

The pagewidth can be set. The default value is 80 columns, set upon initialization.

formfeed tFF

A top of form can be sent to the output device.

indenting tTn or tTn or tT-n

Two kinds of indenting are available, absolute and relative.

contd.

TB sets the value for the number of spaces each line (including its leading blanks) is to be indented. This value remains until it is set again by TB. Negative values of n are converted to 0. The default value for indenting is 0.

TI indents only the line following it. It indents n spaces relative to the value set by TB.

padding ↑P↑ and ↑P-

↑P and ↑P- toggle padding, '↑' for on and '↑-' for off. The name of the current input file and the current page are printed at the top of the page. The default value for padding is off.

double spacing ↑D↑ and ↑D-

↑D and ↑D- toggle the double spacing feature. The default value is off.

multiple spacing ↑SPn

Double spacing is a special case of multiple spacing. Here 'n' is the number of lines to skip between text lines so n=1 is equivalent to double spacing.

centering ↑C↑ and ↑C-

These commands toggle centering. Lines are centered within the current page width. Lines longer than the current page width are written as received. Centering has precedence over right justification; if centering is on, lines will not be justified. The default for centering is off.

right justification ↑J↑ and ↑J-

Lines (and their leading blanks) can be right justified within the current page width. Lines longer than the current width are written as received. The default for justification is on.

include files ↑IN↑string

Files can be 'included' as in the compiler so that input will be taken from the file named in 'string'. Calls to include can be nested arbitrarily up to 16 deep.

The Main Loops and the Stack

After initializing and prompting the user for the name of the source and destination files, the program enters the two main, nested loops:

```
REPEAT
  stackpointer:=POP(stackpointer,...);
  WHILE NOT eof(sourcefile) DO
  BEGIN
    readsource(line,sourcelines);
    IF command(line)
    THEN interpret(line)
    ELSE BEGIN
      fix(line);
      writedest(line,...);
    END;
  END;
UNTIL done;
```

Upon entering the REPEAT loop, the program POPS off the stack the name of the source file obtained by the procedure 'getsourcefile' which put it there in the first place. The stack and the function 'POP' manage the possibly nested calls to the formatter's 'include file' command.

The stack is an ARRAY[1..maxstack] OF

```
TYPE
  stackentry=RECORD
    name:filename;
    linepointer:INTEGER;
  END;
```

where 'name' is the name of a file and line pointer is the number of lines which have been read from the file.

The function 'POP', POPS an entry off the stack. It takes as input the current value of 'stackpointer' and returns as function result the new value of 'stackpointer'; it also returns the name of the file to be used as input and the number of lines which have been read from the file. It makes a call to 'openfile', which opens the input file, and it does dummy reads from the file so that the next line to be read will be linepointer+1.

At this point the program enters the inner loop where it keeps on reading from the source file until the

contd.

end of the file or until an include file command is encountered. More on the latter below.

The procedure 'readsource' isolates the task of reading input. It traps fatal i-o errors so as to permit graceful exits from the program. It also updates 'sourcelines' which is the number of lines which have been read from the source file.

The function 'command' is BOOLEAN valued. It merely checks to see whether the current line is a command line or a line of text.

If a command line is found it is interpreted. First the line is capitalized and all blanks are removed. Then the command mnemonic is looked up. If the mnemonic is recognized, appropriate action, ranging from setting a toggle to including a file, is taken.

If 'command' returns false, then the line must be a line of text. In this case the line is 'fixed' then sent to the output file. 'fix' and 'writedest' together do all of the formatting.

'fix' performs those formatting functions which need to be done before output: centering, indenting and justification. Since I thought these string operations might be generally useful I decided to isolate them in a UNIT. More on this below.

'writedest' outputs the fixed up line to the destination file. It also keeps track of the number of lines sent to the destination file so that it knows when to paginate.

The inner loop is executed until the end of the source file is encountered. However the actual file used as source can change through use of the include command. See the procedure 'include' (part of 'interpret') and the functions 'push' and 'pop'.

After exiting the inner, WHILE loop the value of the stackpointer is tested against zero. If there is something in the stack the program goes to the top of the REPEAT loop, pops off a stackentry and starts reading and writing again.

Error Handling =====

Two error handling routines are included, 'error' and 'fatalerror'.

'error' will trap most i-o errors, those errors for which the built-in function 'ioresult' returns a non-zero value. The routine lets the programmer continue with execution after attempting to fix the error.

A good example of how 'error' is used occurs in 'pop'. An entry is popped off the stack and attempt to open the corresponding file is made. If the file can't be found or opened 'error' takes over and the user has a shot at rectifying the situation, say by inserting the diskette which contains the needed file.

'fatalerror' handles fatal errors, errors which simply shouldn't happen! The main function of the routine is to trap fatal read and write errors in the procedures 'readsource' and 'writedest'. The routine provides clean exits from 'tformat', including closing open files, and gives some information on what happened. It is also used in 'pop' and 'push' to check for stack underflow and overflow, respectively. Underflow is actually impossible; overflow will occur if the nesting of include files is greater than 16.

UNIT stringstuff =====

'tformat' includes routines for centering, adjusting and indenting lines. Because these might be generally useful I isolated them in a library unit along with a few other string utilities. The result was UNIT stringstuff.

The INTERFACE of 'stringstuff' consists of one type declaration for strings of maximum length and three routines in addition to those mentioned above, namely 'capitalize', 'fsrint' and 'flushblanks'.

'capitalize' does what its name implies: it capitalizes lower case alphabets.

'flushblanks' removes all spaces from a string.

contd.

'fstrint' is a function which converts numeric strings to integers. It does not check for integer overflow and keeps converting until it reaches the end of the string or finds a non-numeric character.

The IMPLEMENTATION portion of the UNIT has one procedure, 'fillblanks', which is used by the rest of the UNIT. It inserts a specified number of blanks, 'count', at the position 'pos' of the target string 'line'.

Note that 'center', 'justify' and 'indent' each will typically lengthen the source string passed to them. It is the programmer's responsibility to insure that the routines will not lengthen the source string beyond the length declared for it in the host program. The easiest way to do this is to use strings of length 255 in the host program. If you use strings of lesser length you will have to invoke the V- compiler option when using the UNIT.

Happy computing and...

BLAISE AWAY!

```
(*$U-*)
(*$S+*)
PROGRAM tformat;
USES (*$USTRINSTUFF.CODE*) stringstuff;
CONST
  maxstrlength=255;
  namelength=31;
  paselength=60;
  mincomlen=3; (* MINIMUM length for a command line embedded in source file *)
  cstrnlen=2; (* source file command mnemonics are 2 characters long *)
  done=FALSE;
  commprefix='^'; (* first character in a command line *)
  escape=27; (* decimal value of escape character *)
  bell=7; (* decimal value of bell character *)
  maxstack=16; (* maximum number of include files *)
```

```
TYPE
  direction=(dread,dwrite);(* used by 'openfile' to specify read or write file *)
  filename=STRING[namelength];
  textfile=TEXT;
  fataltypes=(funknown,freadingssource,fwritingdest,fstackfull,funderflow);
  (* internal mnemonics for formatting commands *)
  comctype=(cnull,cinclude,cformfeed,casinate,cnopasinter,
    cdblspc,cnodblspc,cmultispac,ctab,ccenter,cnocenter,
    cspawidth,ctemptab,cjustify,cnojustify,cunknown);
  (* external mnemonics for formatting commands *)
  commstring=STRING[cstrnlen];
  stackrange=0..maxstack;
  stackentry=RECORD
    name:filename;
    linepointer:INTEGER;
  END;
```

MS.SPELLER

Apple II
UCSD Pascal

Compatible with TV or 80 column monitors

only 60.00

plus 2.50 shipping
Va. residents add 4% tax

spelling correction

includes text formatter

dictionary with 2000 hard to spell words

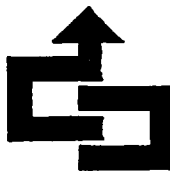
free with purchase of ms.speller

user-created dictionary with options to add, correct, or skip a word while processing a text utilities: list

provides complete word processing capability with UCSD editor Pascal program listings documents letters

- o entire dictionary
- o user added words
- o words added in current session

interactive query updating



Intelligent Computer
Systems Corporation

722 South 24th St.
Arlington, Va. 22202
703 684-7389

VAR

```
destname,sourcename:filename;  
destfile,sourcefile:textfile;  
destlines,sourcelines,currentpage:INTEGER;  
line:maxstring;  
comtable:ARRAY[comtype] OF comstring;  
numberins:BOOLEAN; (* toggle for pagination *)  
centerins:BOOLEAN; (* toggle for centering *)  
dojustify:BOOLEAN; (* toggle for line justification *)  
pagewidth,skipcount,tabcount,temptabcount:INTEGER;  
ioerror:INTEGER; (* used to store result of 'ioresult' *)  
stackpointer:stackrange;  
stack:ARRAY[1..maxstack] OF stackentry;
```

PROCEDURE globalinit;

BEGIN

```
comtable[cnull]:=' ';  
comtable[cinclude]:=' IN';  
comtable[cformfeed]:=' FF';  
comtable[cpasinate]:=' P+';  
comtable[cnopasinate]:=' P-';  
comtable[cdblspc]:=' D+';  
comtable[cnodblspc]:=' D-';  
comtable[ctab]:=' TB';  
comtable[ctemptab]:=' TT';  
comtable[cmultispace]:=' SP';  
comtable[ccenter]:=' C+';  
comtable[cnocenter]:=' C-';  
comtable[cpagewidth]:=' FW';  
comtable[cjustify]:=' J+';  
comtable[cnojustify]:=' J-';  
comtable[cunknown]:=' ';
```

END;

PROCEDURE initialize(VAR sp:stackrange;

VAR sourcelines,destlines,currentpage:INTEGER);

BEGIN

```
sp:=0;  
sourcelines:=0;  
destlines:=0;  
currentpage:=1;  
  
sourcename:='';  
destname:='';  
line:='';  
skipcount:=0;  
tabcount:=0;  
temptabcount:=0;  
numberins:=FALSE;  
centerins:=FALSE;  
dojustify:=TRUE;  
pagewidth:=90;
```

END;

PROCEDURE error(mssg:STRING;ioerror:INTEGER);

VAR

ch:CHAR;

BEGIN

```
page(output);  
gotoxy(0,5);  
writeln(chr(bell),'Error ',mssg,'. ioresult=',ioerror);  
writeln;  
write('Type <ESC> to abort, any other character to continue. >> ');
```

contd.

```

read(ch);
writeln;
IF ch=chr(escape) THEN
  BEGIN
    (**I-*)
    close(sourcefile,lock);
    close(destfile,lock);
    (**I+*)
    exit(program);
  END;
END;

```

```

PROCEDURE fatalerror(ferror:fataltypes;ioerror:INTEGER);
BEGIN
  writeln(chr(bell),chr(bell));
  CASE ferror OF
    freadingsource:writeln('Error reading source file. ioresult=',ioerror);
    fwritingdest:writeln('Error writing destination file. ioresult=',ioerror);
    fstackfull:writeln('Error. Include stack full. ');
    funderflow:writeln('Error. Include stack underflow. ');
  END;
  (**I-*)
  close(sourcefile,lock);
  close(destfile,lock);
  (**I+*)
  writeln('Program aborted. ');
  exit(program);
END;

```

```

FUNCTION openfile(d:direction;VAR tf:textfile;VAR title:filename;
                 VAR ioerror:INTEGER):BOOLEAN;

```

```

CONST
  titlen=18;
VAR
  heap:↑INTEGER;

BEGIN
  (* zero the system's directory buffer by marking the heap *)
  mark(heap);
  IF length(title)>titlen THEN title:=copy(title,1,titlen);
  capitalize(title);
  IF (pos('.TEXT',title)=0) THEN title:=concat(title, '.TEXT');
  (**I-*)
  IF d=dread THEN reset(tf,title)
  ELSE IF title='CONSOLE:' THEN reset(destfile,title)
  ELSE rewrite(tf,title);
  (**I+*)
  ioerror:=ioresult;
  openfile:=ioerror=0;
END;

```

```

FUNCTION POP(sp:stackrange;
           VAR sourcename:filename;VAR sourcelines:INTEGER):stackrange;

```

```

VAR
  i:INTEGER;
BEGIN
  IF sp=0 THEN fatalerror(funderflow,ioerror)
  ELSE BEGIN
    close(sourcefile,lock);
    WITH stack[sp] DO
      BEGIN
        sourcename:=name;
        sourcelines:=linepointer;
      END;

```

```

SP:=SP-1;
REPEAT
  IF NOT openfile(dread,sourcefile,sourcename,ioerror)
    THEN error(concat('opening ',sourcename),ioerror);
UNTIL ioerror=0;
i:=0;
WHILE i<sourcelines DO BEGIN
  readln(sourcefile,line);
  i:=i+1;
END;
POP:=SP;
END; (* IF SP=0... *)
END;

FUNCTION push(sp:stackrange;
              sourcename:filename;sourcelines:INTEGER):stackrange;
BEGIN
  IF SP=maxstack THEN fatalerror(fstackfull,ioerror)
  ELSE BEGIN
    SP:=SP+1;
    WITH stack[sp] DO
      BEGIN
        name:=sourcename;
        linepointer:=sourcelines;
      END;
    push:=sp;
  END; (* IF SP=maxstack... *)
END;

PROCEDURE setsourcefile(VAR sourcename:filename;VAR sp:stackrange);
CONST
  syswrk='*SYSTEM.WRK.TEXT';
VAR
  ch:CHAR;
BEGIN
  page(output);
  gotoxy(0,5);
  REPEAT
    writeln('Select source file name:');
    writeln;
    writeln('  S)YSTEM.WRK.TEXT');
    writeln;
    writeln('  D)isk file');
    writeln;
    writeln('  Q)uit program');
    writeln;
    write('What''s it going to be? ');
    read(keyboard,ch);
    WHILE NOT (ch IN ['S','s','D','d','Q','q']) DO
      BEGIN
        write(chr(7));
        read(keyboard,ch);
      END;
    IF ch IN ['S','s'] THEN sourcename:=syswrk
    ELSE IF ch IN ['q','Q'] THEN exit(program)
    ELSE BEGIN
      writeln;
      writeln;
      write('Enter file name: ');
      readln(sourcename);
    END;
  REPEAT

```

```

IF NOT openfile(dread,sourcefile,sourcename,ioerror)
  THEN error(concat('openins ',sourcename),ioerror);
UNTIL ioerror=0;
stackpointer:=push(sp,sourcename,sourcelines);
END;

```

```

PROCEDURE setdestfile(VAR destname:filename);

```

```

VAR
  ch:CHAR;
BEGIN
  page(output);
  gotoxy(0,5);
  REPEAT
    writeln('Select destination file name. ');
    writeln;
    writeln('  P)RINTER: ');
    writeln;
    writeln('  C)ONSOLE: ');
    writeln;
    writeln('  D)isk file ');
    writeln;
    writeln('  Q)uit program. ');
    writeln;
    write('What''s it going to be? ');
    read(keyboard,ch);
    WHILE NOT (ch IN ['P','p','C','c','D','d','Q','q']) DO
      BEGIN
        write(chr(7));
        read(keyboard,ch);
      END;
    IF ch IN ['P','p'] THEN destname:='PRINTER: '
    ELSE IF ch IN ['C','c'] THEN destname:='CONSOLE: '
    ELSE IF ch IN ['Q','q'] THEN exit(program)
    ELSE BEGIN
      writeln;
      writeln;
      write('Enter file name: ');
      readln(destname);
      END;
    IF NOT openfile(dwrite,destfile,destname,ioerror)
      THEN error(concat('openins ',destname),ioerror);
    UNTIL ioerror=0;
  END;

```

```

PROCEDURE readsource(VAR line:maxstring;VAR sourcelines:INTEGER);

```

```

BEGIN
  (*I-*)
  readln(sourcefile,line);
  ioerror:=ioresult;
  IF ioerror<>0 THEN fatalerror(freadingssource,ioerror)
  ELSE sourcelines:=sourcelines+1;
  (*I+*)
END;

```

```

PROCEDURE pascinate(VAR linecount,pasecount:INTEGER);

```

```

BEGIN
  writeln(destfile,'FILE: ',sourcename,' ':(namelength+1-length(sourcename)),
    'page number ',pasecount:4);
  writeln(destfile);
  linecount:=2;
END;

```

```

PROCEDURE formfeed(VAR linecount,pasecount:INTEGER);
BEGIN
  pase(destfile);
  pasecount:=pasecount+1;
  linecount:=0;
  IF numberins THEN pasinate(linecount,pasecount);
END;

```

```

PROCEDURE fix(VAR line:maxstring);
BEGIN
  IF length(line)>0 THEN
    BEGIN
      IF scan(length(line),<>' ',line[1])=length(line)
        THEN line:=''
        ELSE
          BEGIN
            IF (temptabcount+tabcount IN [1..maxstringlength])
              THEN indent(line,temptabcount+tabcount);
            IF centerins THEN center(line,pasewidth)
              ELSE IF dojustify THEN justify(line,pasewidth);
          END;
        END;
    END;
END;

```

```

PROCEDURE writedest(line:maxstring;VAR linecount,pasecount:INTEGER);
VAR
  i:INTEGER;

```

```

FUNCTION bumplinecount(linecount:INTEGER):INTEGER;
BEGIN
  linecount:=linecount+1;
  IF linecount=paselength THEN
    BEGIN
      linecount:=0;
      pasecount:=pasecount+1;
      IF numberins THEN pasinate(linecount,pasecount);
    END;
  bumplinecount:=linecount;
END;

```

```

BEGIN
  (*$I-*)
  writeln(destfile,line);
  (*$I+*)
  ioerror:=ioresult;
  IF ioerror<>0 THEN fatalerror(fwritingdest,ioerror)
  ELSE BEGIN
    linecount:=bumplinecount(linecount);
    IF NOT ((destname='PRINTER:.TEXT') OR (destname='CONSOLE:.TEXT')) THEN
      IF (linecount MOD 40)=0 THEN writeln
        ELSE write(' ');
    i:=0;
    WHILE (i<skipcount) AND (linecount>0) DO
      BEGIN
        writeln(destfile);
        linecount:=bumplinecount(linecount);
        i:=i+1;
      END;
    END; (* IF ioerror... *)
  END; (* writedest *)

```

```

FUNCTION command(line:maxstring):BOOLEAN;

```



```

BEGIN
  IF length(line)<mincomlen THEN command:=FALSE
  ELSE command:=line[1]=commprefix;
END;

PROCEDURE interpret(line:maxstring);
VAR
  cstring:commstring;
  cmd:comtype;

PROCEDURE include(line:maxstring);
BEGIN
  stackpointer:=push(stackpointer,sourcename,sourcelines);
  close(sourcefile,lock);
  sourcename:=line;
  REPEAT
    IF NOT openfile(dread,sourcefile,sourcename,ioerror)
    THEN error(concat('opening new source file ',sourcename),ioerror);
  UNTIL ioerror=0;
END;

BEGIN
  (* capitalize the line *)
  capitalize(line);
  (* remove blanks *)
  flushblanks(line);
  (* set the two character mnemonic for the command *)
  cstring:=copy(line,2,2);
  (* delete the escape-to-command character and the mnemonic from 'line' *)
  delete(line,1,3);
  (* look up the command *)
  cmd:=cnull;
  REPEAT cmd:=succ(cmd)
  UNTIL (cstring=comtable[cmd]) OR (cmd=cunknown);
  (* execute the command *)
  CASE cmd OF
    cinclude:include(line);
    cformfeed:formfeed(destlines,currentpage);
    cpasinate:BEGIN
      numbering:=TRUE;
      IF destlines=0 THEN pasinate(destlines,currentpage);
    END;
    cnopasinate:numbering:=FALSE;
    cdblspc:skipcount:=1;
    cnodblspc:skipcount:=0;
    ctabs:BEGIN
      tabcount:=fstring(line);
      IF tabcount<0 THEN tabcount:=0;
    END;
    ctemptab:temptabcount:=fstring(line);
    cmultispace:BEGIN
      skipcount:=fstring(line);
      IF skipcount<0 THEN skipcount:=0;
    END;
    ccenter:centering:=TRUE;
    cnocenter:centering:=FALSE;
    cpasewidth:BEGIN
      pasewidth:=fstring(line);
      IF pasewidth<0 THEN pasewidth:=80;
    END;
    cjustify:dojustify:=TRUE;
    cnojustify:dojustify:=FALSE;

```

```

cnull,cunknown;;
END;
END;

BEGIN
globalinit;
REPEAT
  initialize(stackpointer,sourcelines,destlines,currentpage);
  setsourcefile(sourcenam,stackpointer);
  setdestfile(destname);
  REPEAT
    stackpointer:=POP(stackpointer,sourcenam,sourcelines);
    WHILE NOT eof(sourcefile) DO
      BEGIN
        readsource(line,sourcelines);
        IF command(line) THEN interpret(line)
        ELSE BEGIN fix(line);writedest(line,destlines,currentpage);END;
      END;
    UNTIL stackpointer=0;
    close(sourcefile,lock);
    close(destfile,lock);
  UNTIL done;
END.
(*$S+*)
UNIT stringstuff;

INTERFACE
TYPE
  maxstring=STRING[255];

PROCEDURE capitalize(VAR line:maxstring);
PROCEDURE flushblanks(VAR line:maxstring);
FUNCTION fstrint(line:maxstring):INTEGER;
PROCEDURE center(VAR line:maxstring;width:INTEGER);
PROCEDURE justify(VAR line:maxstring;width:INTEGER);
PROCEDURE indent(VAR line:maxstring;count:INTEGER);

IMPLEMENTATION
CONST
  maxstrlen=255;
TYPE
  byte=0..maxstrlen;
VAR
  index:byte;

PROCEDURE fillblanks(VAR line:maxstring;pos,count:byte);
BEGIN
  (*$R-*)
  moveright(line[pos],line[pos+count],length(line));
  fillchar(line[pos],count,' ');
  line[0]:=chr(length(line)+count);
  (*$R+*)
END;

PROCEDURE capitalize;
CONST
  ordsmla=97;
  ordsmlz=122;
  shiftcase=32;
BEGIN
  FOR index:=1 TO length(line) DO
    IF line[index] IN [chr(ordsmla)..chr(ordsmlz)] 32

```

contd.

```

THEN line[index]:=chr(ord(line[index])-shiftcase);
END;

```

```

PROCEDURE flushblanks;

```

```

BEGIN
IF length(line)=0 THEN exit(flushblanks)
ELSE REPEAT
index:=scan(length(line),' ',line[1]);
IF index<length(line) THEN delete(line,index+1,1);
UNTIL index=length(line);
END;

```

```

FUNCTION fstrint;

```

```

VAR
sign:BOOLEAN;
x:INTEGER;
BEGIN
IF length(line)=0 THEN BEGIN fstrint:=0;exit(fstrint); END
ELSE IF line[1]='-' THEN BEGIN sign:=TRUE;delete(line,1,1); END
ELSE BEGIN sign:=FALSE;IF line[1]='+' THEN delete(line,1,1); END;
x:=0;

```

```

WHILE length(line)>0 DO

```

```

IF line[1] IN ['0'..'9'] THEN
BEGIN x:=10*x+ord(line[1])-ord('0');delete(line,1,1); END
ELSE line:='';
IF sign THEN fstrint:=-x
ELSE fstrint:=x;
END;

```

```

PROCEDURE center;

```

```

BEGIN
IF length(line)>width THEN exit(center)
ELSE BEGIN
index:=(width-length(line)) DIV 2;
fillblanks(line,1,index);
END;
END;

```

```

PROCEDURE justify;

```

```

VAR
widenings,pos,blanksneeded,nmrofsaps,
leftmost,rightmost:0..maxstrlen;
BEGIN
IF length(line)>width THEN exit(justify);
leftmost:=scan(length(line),<>' ',line[1])+1;
rightmost:=scan(-length(line),<>' ',line[length(line)]+length(line));
IF leftmost<=rightmost THEN
BEGIN
nmrofsaps:=0;
pos:=leftmost;
pos:=scan(rightmost-pos,' ',line[pos])+pos;
WHILE pos<rightmost DO
BEGIN
nmrofsaps:=succ(nmrofsaps);
pos:=pos+1;
pos:=scan(rightmost-pos,' ',line[pos])+pos;
END;
blanksneeded:=width-rightmost;
pos:=leftmost;
pos:=scan(rightmost-pos,' ',line[pos])+pos;
WHILE (pos<rightmost) AND (nmrofsaps>0) DO
BEGIN

```

```

widenings:=blanksneeded DIV nmrofsaps;

```

```

fillblanks(line,pos,widenings);

```

```

blanksneeded:=blanksneeded-widenings;

```

```

nmrofsaps:=nmrofsaps-1;

```

```

pos:=pos+widenings+1;

```

```

rightmost:=rightmost+widenings;

```

```

pos:=scan(rightmost-pos,' ',line[pos])+pos;

```

```

END;

```

```

END; (*IF leftmost<=rightmost... *)

```

```

END;

```

```

PROCEDURE indent;

```

```

BEGIN

```

```

IF (length(line)+count>maxstrlen) THEN exit(indent)

```

```

ELSE fillblanks(line,1,count);

```

```

END;

```

```

BEGIN

```

```

END.

```

A REVIEW OF LIBRARY DISK VOLUME 32 GAMES 9

by David C. Stern

I have followed the review format initiated by Brian Dormer in the August 1980 issue of WAP. A brief review of each game on the disk is provided below. I have also modified the individual listings of the games on Disk Volume 32 to provide instructions on running the more complex games. These instructions are either called up at the beginning of the game or as "REM" statements in the program listings.

As in Brian's review, the rating scale is as follows:

***** Superb
**** Better than average
*** Good
** OK
* Forget it.

SPACE FLIGHT - Try to pilot a space ship from one end of the galaxy to the other. You do this by firing either your main or "half" rockets. This game does not use graphics (but it sure could use a graphic plot) and is hard to understand.
Rating: ***

MONSTER CHASE - Try to avoid being eaten by the monster. You can move north, east, south and west or stay still, and you must survive 10 moves without being eaten by the monster. Easy, once you try it a few times. Text gameboard. Fun for kids in third or fourth grade. Rating: **

CHRISTMAS TREE - Hi-res tree is displayed. You can place up to 200 lights by using the game paddles. Choose from 5 colors and make them flash or glow steadily. Color monitor is a must. Rating: ***

LOW SCORE II - Nice hi-res game. The object is to get a lower score than the opponent. You can play against the APPLE or another person. Rating: ****

INTERNA-MAZE - The APPLE constructs a lo-res maze and actually puts you inside the maze. You can see the corridor ahead of you (sort of a rat's eye view of running a maze). You can turn right or left, go forward or backward. An excellent example of lo-res graphics. Hard, but fun. Rating: ****

AT-THE-TRACK - A lo-res horse race game. Up to six people can play as you bet on the horses. No skill, just chance. Horses move very slowly, but graphics are good. Rating: **

JUGGLE - You control the paddle at the bottom of the screen. Try to "juggle" the balls by letting them bounce off the paddle. Add more balls and the game gets increasingly more challenging and each "juggle" is worth more points. Keeps a high score but does not save it to the disk. Rating: ***

JIG-SAW PUZZLE - Try to put the pieces of this lo-res puzzle together. 9 pieces are randomly placed at the start of the game. After each move, the APPLE tells you how many pieces are still in the wrong place.
Rating: ***

STRATOLASER - The object of this game is to destroy Klukon fighters. The APPLE displays a plotting board. You can launch probes either horizontally or vertically, gain information on the position of the target, and attempt to destroy it with your stratolaser. Interesting but slow. I would rather play Cyber-Strike.
Rating: ***

HIRES PONG - A revised version of Penny Arcade by Bill Budge. No games are added, but there are more instructions than on the tape version. Hi-res. Rating: ****

WORD DOODLES - A good example of the use of "Higher Text". In this game you guess what the phrase or symbol on the screen means. To get a list of the answers, you must write to Washington Apple Pi. Hi-res. Good if you like word doodles.
Rating: ***

MAZE MAKER - A very quick maze-making program. You can set the width and length. It first draws the maze in lo-res, then prints it out in text for a printer. Rating: ***

ROADRUNNER - A math game in text. It displays 50 add/subtract problems, one at a time. You have 20 seconds to solve each problem. After you get one wrong, it rates you from snail (lowest) to roadrunner (highest). No graphics. WAP Vol. 18 has better math games. Rating: *

SPELUNKER II - An adventure game. You are at the mouth of a cave. Can you find the hidden treasure? Does not use graphics, but has a nice display in text.
Rating: **

SPACEMAZE II - A revised version of the original Space Maze (WAP Vol. 15, Games 7). There are four skill levels. Your object is to pilot your space ship through the maze to reach the dilithium crystals at the end. Hi-res. Rating: *****

(Editor's Note: David, age 11, is an active member of SIGAMES. He will be graduating from Luxmanor Elementary School, in North Bethesda, later this Spring. Versions of Disk Volume 32 with David's instructions will be available soon.)



CLASSIFIEDS

WANTED TO RENT: APPLE II with disk, FP and INT, DOS 3.3. Will pay \$5 per hour cash for occasional sessions of 1 - 3 hours each (your place and your convenience). U/L-case characters and printer desired, but not required. Ernie Brown, WAP#618, 763-7686 (W), 881-0115 (H).

WANTED TO RENT: DB Master, for one week. Bill Etue, (703) 620-2103.

FOR SALE: Micromodem - excellent documentation and some applications software. \$325 or best offer. Call Alan at 946-2585 and leave message.

FOR SALE: Disk drive controller (3.3), \$100; Centronics 730 printer, \$395. Ira Cotton (WAP#244), 468-2266 (H), 951-2693 (W).

ADVERTISING RATES

The following table shows our current advertising rates. Our newsletter distribution is about 1250 copies, with approximately 200 of these going around the country. If you would like to advertise please send camera ready ad copy (half tones are permitted, no bleeds) by the 10th of the month to our PO Box. If you cannot prepare camera ready copy send us your specifications. Our Art Department will be happy to assist you.

	RATES			
	FULL PAGE	HALF PAGE	QTR PAGE	8TH PAGE
Single issue (or split series)	\$ 40	\$ 25	\$ 15	\$ 8
3-months series (or more)	35	20	10	5
Full-year contract	30	15	10	5

★ ★ ★ STOCK MARKET UTILITIES ★ ★ ★ 4 STOCK MARKET PROGRAMS ON DISK

Four programs provide a complete programming system for entry and storage of stock data, data correction, autoscaling Hi-Res graphical display of performance, and building historical data files electronically (program to download data not included). **Now with HARD COPY GRAPHICS.**

STK 1 (39 Sectors) provides complete utilities for manual entry of stock data. **Features:** names stored alphabetically by exchange, easy addition and deletion of names, automatic prompting and extensive error trapping for data entry (date, volume, price), numerous entry points for data correction, all data displayed prior to updating stock files with further option for data correction, input historical data to a single data file, display contents of individual stock files from disk, option to reduce files to last 260 entries for high-res graphics. All data files are fully accessible.

DATA CORRECTOR (31 Sectors) used to correct and rewrite stock data files. **Features:** option for general data correction - correct any entry, option for stock splits - all prices and volumes prior to split scaled by split ratio (transaction dollars constant) to provide continuous momentum and price curves, also correct for incomplete updating due, for example, to a power outage.

EVAL (22 Sectors) provides comparative evaluation of stock performance. **Features:** synchronizes NYSE index ave with first stock entry, option to evaluate all stocks automatically or just one, simultaneous high-res display of momentum, price, and price relative to NYSE index ave, auto scaling graphics, numerical figure of merit for performance relative to NYSE index ave.

MIROQ (12 Sectors) is used to build historical data files electronically by converting downloaded stock price data obtained from Compuserve's Micro-Quote financial data base to data files compatible with these programs.

Programs written by H. S. PILLOFF. **DOS 3.3**
Requires Apple II™ ROM Applesoft™ 48K and Disk (DOS 3.2)

★ ★ ★ ELECTRONIC STOCK PACKAGE ★ ★ ★

A complete system including password and programs for accessing the Dow Jones Stock Quote Reporter (contains more than 6000 daily stock prices). Current rates permit nightly updating for about 1.5¢ per stock.

Downloading programs provide for auto dialing, logging on, retrieving daily data (prev. close, open, high, low, close, volume) for up to 200 stocks stored in easily edited file, disconnecting from system, and then writing data to a single file on the user's disk. Data can then be displayed or printed.

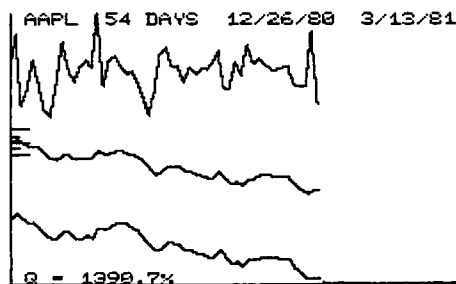
Conversion programs read this disk file, formats data (M/D/Y/VOL/FNL), and automatically updates each individual stock file. Format is fully compatible with STOCK MARKET UTILITY PROGRAMS.

Requires Apple II/II+™, Applesoft™ 48K, Disk (DOS 3.2 or 3.3), and D.C. Hayes Micromodem II™.

Electronic Stock Package (includes Dow Jones password) \$80.00
Stock Market Utility Programs \$59.95

COMING SOON!! Black-Scholes Stock Option Evaluation Programs Specializing in Investment Software

Foreign orders add \$10 for shipping. **H&H SCIENTIFIC** VISA/Mastercard Accepted
13507 Pendleton Street, Oxon Hill, MD 20022 Tel (301) 292-3100
Apple II/II+, and Applesoft are trademarks of Apple Computer, Inc. Micromodem II is a trade mark of D.C. Hayes Assoc., Inc.





APPLE ORCHARD SUBSCRIPTIONS

P.O. BOX 1493 BEAVERTON, OR 97075, USA

The International Apple Core will make individual subscriptions to "The Apple Orchard" available commencing with Volume 1, Number 3.

NAME _____

STREET _____

CITY _____ STATE _____ ZIP _____

COUNTRY _____

Annual Subscription Rate: \$10.00 per year (Four issues - Published Quarterly)

First Class Postage: \$5.00 per year additional (required for Canada, Mexico, APO, and FPO addresses)

Overseas and other foreign air mail postage (required): \$10.00 per year additional

TOTAL REMITTANCE ENCLOSED: \$(USA) _____

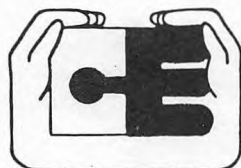
Make check or money order payable to "International Apple Core" and return with this form to:

APPLE ORCHARD SUBSCRIPTIONS

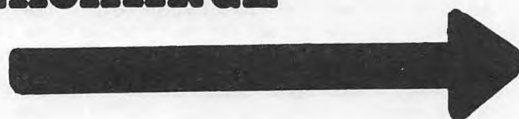
P.O. Box 1493

Beaverton, OR 97075, U.S.A.

1/81



USED COMPUTER EXCHANGE wants to know ...



HOW GOOD A DEAL IS A USED COMPUTER?

Many serious users are finding these days that used computer equipment can be as good a buy -or better- than new. However, buying and selling used equipment can be risky without accurate information about pricing, market trends, maintenance records, etc. (One outfit we contacted buys used equipment, services it, and marks it up 165% !)

The USED COMPUTER EXCHANGE is currently gathering hard data on all key makes and models of used microcomputer systems and peripherals. Help us help you by filling out the following survey. Your experiences combined with our system for matching buyers and sellers nationwide can cut the risks, cost, and effort of making a transaction.

In return for your time, we will publish a summary and analysis of the results in a future issue of this magazine and send details to any participant who checks the appropriate box.

Used Equipment I Have Bought or Sold:

	Make	Model	Description	Bought	Sold	Month & Year	Price	Retail Price Would Have Been
<input type="checkbox"/>	CPU	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	_____	_____	_____
<input type="checkbox"/>	DISK	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	_____	_____	_____
<input type="checkbox"/>	PRINTER	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	_____	_____	_____
<input type="checkbox"/>	MONITOR	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	_____	_____	_____
<input type="checkbox"/>	OTHER	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	_____	_____	_____

I MADE THE TRANSACTION USING: NEWSPAPER CLASSIFIEDS CLUB MEMBERSHIP
 MAGAZINE ADS FRIENDS
 OTHER _____

DURING MY EFFORTS FINDING IT AND COMPLETING THE TRANSACTION, I SPENT _____ HOURS RESEARCHING, READING, TALKING TO FRIENDS, ETC.

WHEN MY EQUIPMENT WAS INSTALLED EVERYTHING WAS FINE I HAD PROBLEMS _____

Maintenance Experience:

I DO ALL SOME NONE OF MY OWN MAINTENANCE.
 A LOT A LITTLE

MAINTENANCE NOT DONE BY ME IS DONE _____ % IN STORES _____ % BY FREE LANCERS
 _____ % FREE BY FRIENDS _____ % BY _____

AT \$20 PER HOUR FOR LABOR I WOULD PLAN A YEARLY MAINTENANCE BUDGET AT THE FOLLOWING PER CENT OF THE ORIGINAL PURCHASE PRICE (includes items bought new):

	Make	Model	Date Bought	0-10%	11-20%	21-30%	Other
CPU	_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
DISK	_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
PRINTER	_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
MONITOR	_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____
OTHER	_____	_____	_____	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	_____

BELOW IS A DESCRIPTION OF THE ITEM I HAVE OWNED WHICH LEAST MET MY EXPECTATIONS AND CAUSED ME THE WORST MAINTENANCE PROBLEMS:

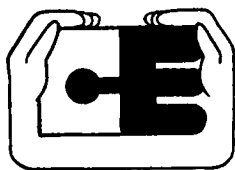
Item	Date Bought	Price	Total Maint. Expenses
PROBLEMS: _____	_____	_____	_____

Other Comments ON USED EQUIPMENT:

PLEASE SEND ME A COPY OF THE DETAILED FINDINGS OF THIS SURVEY.

NAME: _____
 ADDRESS: _____
 PHONE: _____

PLEASE SEND TO:



USED COMPUTER EXCHANGE

2329 HUNTERS WOODS PLAZA, RESTON, VA. 22091

OR CALL: JOAN AT (703) 471-0044

 WASHINGTON APPLE PI
 MAIL ORDER FORM

Washington Apple Pi now has a program library, and disks are available for purchase by anyone. The price to members is \$5.00 per disk and \$8.00 to non-members. These disks are chock full of exceptional programs - the utilities are especially useful. The games are some of the best - not just simple and uninteresting ones. You may pick them up at any meeting or have them mailed for \$2.00 per disk additional. (If you order five or more the additional charge will be \$10.00 total.) They will come in a protective foam diskette mailer.

PROGRAM DISKETTES

Members: \$5.00 picked up at meeting
 \$7.00 mailed to you (for the first five, remainder at \$5.00)

Non-members: \$8.00 per disk picked up at meeting
 \$10.00 mailed to you (for the first five, remainder at \$8.00)

Volume 1	Utilities I	()	Volume 31	Plot Utilies	()
Volume 2	Utilities II	()	Volume 32	Games XI	()
Volume 3	Games I	()	Volume 33	Accounting	()
Volume 4	Games II	()	Volume 34	Solar Tutor	()
Volume 5	Games III	()	Volume 35	Garden Management	()
Volume 6	Games IV	()	Volume 36	Games XII	()
Volume 7	Games V	()	Volume 37	Utilities IX	()
Volume 8	Utilities III	()			
Volume 9	Educational I	()	Volume 100	Dos 3.3 Utilities A	()
Volume 10	Math/Science	()	Volume 180	Dungeon Designer	()
Volume 11	Graphics I	()	Volume 181	Beginner's Cave	()
Volume 12	Games VI	()	*Volume 182	Lair of Minotaur	()
Volume 13	Games	()	*Volume 183	Cave of the Mind	()
Volume 14	IAC Utilities IV	()	*Volume 184	Zyphur Riverventure	()
Volume 15	Games VII	()	*Volume 185	Castle of Doom	()
Volume 16	Utilities V	()	*Volume 186	Death Star	()
Volume 17	Graphics II	()	*Volume 187	Devil's Tomb	()
Volume 18	Educational II	()	*Volume 188	Caves of Treas. Isl.	()
Volume 19	Communications	()	*Volume 189	Furioso	()
Volume 20	Music	()	*Volume 190	The Magic Kingdom	()
Volume 21	Apple Orchard	()	*Volume 191	The Tomb of Molinar	()
Volume 22	Utilities VI	()	*Volume 192	Lost Island of Apple	()
Volume 23	Games VIII	()	Pascal: PIG1:		()
Volume 24	Games IX	()	PIG2:		()
Volume 25	Utilities VII	()	PIG3:		()
Volume 26	Stocks/Investments	()	PIG4: BIOS		()
Volume 27	Math	()	*Vol. 181	required with these	
Volume 28	Planetfinder	()	disks.		
Volume 29	Utilities VIII	()			
Volume 30	Games X	()			

 TOTAL ORDER = \$ -----

Check here if you want these shipped---

NOTE: PLEASE ALLOW 6 - 8 WEEKS FOR MAILING.

NAME -----

ADDRESS -----

CITY, STATE, ZIP -----

TELEPHONE -----

Membership No.(1st three digits after WAP on mailing label) -----

Make checks payable to "Washington Apple Pi"

Send order to: Washington Apple Pi- ATTN: Librarian
 PO Box 34511
 Washington, DC 20034

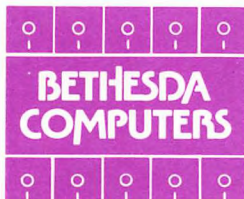
Eve
introduced
the apple to
Adam



Now
we've brought it
to Bethesda

Can we tempt you?

As your authorized Apple dealer, Bethesda Computers specializes in the sale of micro-computers, software, peripheral equipment & accessories. Drop by for a "hands-on" demonstration and professional analysis of your computer needs.



Bethesda Computers
8020 Norfolk Avenue
Bethesda, Maryland 20014
(301) 657-1992