

\$1.50

# Washington Apple Pi



Volume 3

September 1981

Number 8

## Highlights

COUNT SECTORS

PUFF IN

WORD PROCESSOR REVIEW

## In This Issue

	Page
MEMBERSHIP, EVENT QUEUE, EDITORIAL, CLASSIFIEDS.....	1
SIGNEWS.....	2
SIGAME NEWS..... JOHN ALDEN	2
HOTLINE.....	3
TUTORIAL.....	3
PRESIDENT'S CORNER..... DAVID MORGANSTEIN	4
QUESTIONS, QUESTIONS, QUESTIONS..... MARK L. CROSBY	6
A PAGE FROM THE STACK..... DAVID MORGANSTEIN	9
THE MTU HAT..... VANCE GIBONEY	10
COUNT SECTORS FROM APPLESOFT..... WILLIAM SCHULTHEIS	11
WORD PROCESSING SOFTWARE FOR THE APPLE:	
A SYNOPSIS OF THE PUBLISHED REVIEWS.... WALTON FRANCIS	16
THE SYSOP REPLIES..... JOHN L. MOON	22
PUFFIN (DOS TO PASCAL TEXT FILE CONVERSION)..... DR. WO	25
WAP DISK #29 CORRECTION..... BOB SCHMIDT	31
HIT PARADE..... JOHN ALDEN	32



# ComputerLand<sup>®</sup>

We know small computers.



 **apple<sup>®</sup> computer**  
Sales and Service

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

# OFFICERS & STAFF

# EDITORIAL

President -David Morganstein (301) 972-4263  
 Vice President -Mark Crosby (202) 488-1980  
 Treasurer -Dana Schwartz (301) 725-6281  
 Secretary -Jesse Wagstaff (301) 937-4215  
 Members-at-Large -John Moon ABBS Sysop  
 -Nancy Philipp (301) 924-2354  
 -Rich Wasserstrom (703) 893-9147  
 Immed. Past Pres. -Bernard Urban (301) 229-3458  
 Editor -Bernard Urban (above)  
 Associate Editor -Rich Wasserstrom (above)  
 -Genevie Urban (301) 229-3458

Librarians:  
 Disk Mail Order -Sarah Lavilla (301) 926-6355  
 Disk Pick-up -Bill Bowie (301) 924-3455  
 Group Purchases -Dick Elkins (301) 384-9297  
 Advertising -Howard Lefkowitz (301) 649-3373  
 Membership -Bob Peck (301) 621-2719  
 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  
 Bethesda, Md. 20817  
 (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.

Members who would like to sign onto the Washington Apple Pi ABBS system should call the club phone and leave your name (first and last), WAP number and phone number. You will be assigned a password and the message will be forwarded to John Moon who will take care of signing you on.

## 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.) The August meeting is on the 22nd and the September meeting on the 26th.

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.

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.

We received a raft of top-notch articles over the past several weeks, which we will publish this month and next.

As promised, this issue contains "Puffin", by Dr. Wo, the companion article to Dana Schwartz's "Huffin" which appeared last month. "Puffin" converts DOS text files to Pascal files. Dr. Wo suggested that we sub-title the article "Blaise Goes Both Ways", but this is a family magazine!

Walt Francis' synopsis of Apple word processor reviews is probably the most comprehensive article of its kind. Take Walt's advice; "try before you buy".

Bill Schultheis' excellent "Count Sectors in Applesoft" is really a mini-tutorial in problem solving strategies and assembly language programming.

John Moon's "The Sysop Replies" gives all you ABBS users and potential users all you ever wanted to know about the WAP bulletin board. John, ever the gentlemen, did not, however, reveal the name of the well known WAP'er (!!!) who holds the all-time indoor and outdoor record for the most monumental ABBS system crash!

NEXT MONTH--Just to whet your appetite, we have a number of fine articles on tap for the October issue. Ed Knepley will tell us how to get a second HIRES screen for Language Card users and other Pascal graphics tricks. Fred Schulz reports on two products which allow Apple II RAM expansion beyond 64K. Would you believe one megabyte? We still have plenty of space left, so keep those articles coming.

VOLUNTEERS--Take a look at our new "hotline helpers"! These kind folks responded to Boris Levine's questionnaire distributed at the July meeting. Please respect the telephone call restrictions where listed and, in any event, no calls after 10:00 pm. Thanks, Boris for getting this important project moving. And thank you volunteers for sharing your expertise and time. Now, that's what this club is all about!

RSW

## CLASSIFIEDS

For Sale: Epson MX-80 printer w/friction feed and Epson Apple interface. Perfect condition. Am buying a bigger printer. \$650. Kevin Duffy (202) 363-6245.

For Sale: Apple Centronics parallel interface card w/cable. \$100 or best offer. Wally Estes 272-5717 (h), 443-4809 (w).

For Sale: Integer firmware board w/manual. \$165. John Alden (202) 686-1656.

For Sale: Apple Comm. Card. \$125; AJ acoustic coupler. \$125. Hersch Pilloff 292-3100.

# SIG-NEWS

# SIGAME NEWS

by John Alden

SIGAMES is the special interest group of computer hobbyists interested in using their Apples for entertainment.

This month's newsletter features two new regular SIGAMES columns: HIT PARADE and SIGAMES NEWS, both by John Alden. HIT PARADE is SIGAMES' new buyer's guide to games. Each month a new group of games will be featured. This month's feature is graphic adventure games.

SIGAME NEWS will present the agenda for the current month's SIGAMES meeting, the next month's agenda, a synopsis of the prior month's meeting, and a review of one or two new games.

A special feature of this month's SIGAMES meeting will be the presentation of awards by Jim Eatherly to the winners of the game contest.

---

FIG, 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 immediately after the regular meeting of Washington Apple Pi.

---

NEWSIG will meet just after the regular Washington Apple Pi meeting. 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

Welcome to a new feature of the Washington Apple Pi SIGAMES group! This column will be devoted to three tasks: To present information about upcoming SIGAMES events; to present a synopsis of the past month's meeting; and to review new and favorite games.

The next few months promise to be especially exciting at the SIGAMES meetings. Each meeting will begin by surveying a selection of games. New games will be presented and demonstrated on an Apple. Then special guests and speakers will present new and exciting aspects of games. This month, Jim Eatherly will announce the winners of the SIGAMES game contest.

-----

At the September meeting Theron Fuller, President of NOVAPPLES, will introduce speakers on the subject of artificial intelligence. Theron has scheduled speakers who will discuss and demonstrate how artificial intelligence relates to designing games. (There is no truth to the rumor that artificial intelligence refers to persons who have played Space Invaders for the last two years.)

-----

At the last meeting of SIGAMES, several graphic adventure games were reviewed and demonstrated. The games included 'Beneath Apple Monor', 'The Wizard and The Princess', 'Odyssey', 'Akalabeth', and 'Ultima'. These games were selected to illustrate the development of the graphic adventure game.

Tom Lucas was the featured speaker. Tom discussed the development of the various games, how to approach adventures, and

cont'd on p.34

# WAP HOTLINE

Have a problem? The following club members have agreed to help. PLEASE, respect all telephone restrictions, where listed, and no calls after 10:00 pm.

General	Ben Acton	972-1533
	Robert Fretwell	971-2621
	Dave Harvey	527-2704
	Tom Jones	460-8773
	Robert Martin	498-6074
Operating Systems		
Apple DOS	Richard Untied	241-8678 (wknds. only)
CP/M	Robert Fretwell	971-2621
Languages (A=Applesoft, I=Integer, P=Pascal, M=Machine)		
A,I	Jeff Dillon	422-6458
A,I	Tom Jones	460-8773
A	Mark Pankin	370-9219
A,I,P,M	Bill Schultheis	538-4575 (ex.Tue. & Thur)
A,I,M	Richard Untied	241-8678
P	Robert Fretwell	971-2621
Printers	Walt Francis	966-5742
Word Proc	" "	" "
	Ben Acton	972-1533
Visicalc	" "	" "
	Walt Francis	966-5742
Time-Sharing	Chuck Reinbrecht	299-6810
	Dave Harvey	527-2704
Graphics	Bill Schultheis	538-4575 (ex.Tue. & Thur)
Games	Jim Etherly	232-6046

## TUTORIAL

Just as soon as our meeting location settles down, WAP will offer the first of what we hope will be a series of programming tutorials. The first course is designed to take the new user beyond the Applesoft and Integer Basic programming tutorials through four 1 1/2 hour sessions conducted by Dave Morganstein. Enrollment will be limited to assure a student to Apple ratio of 2:1. We hope to start this Fall. Stay tuned to this space for more details. A course outline follows:

### Session I--Introduction

- A. Binary/hex numbers
- B. Bits, bytes, and nibbles
- C. Ram, Rom and devices

### Session II

- A. Memory map: what's really in there
- B. The monitor: examine, move, dis-assemble, step, trace & verify

### Session III

- A. Basic programming
- B. Commands and applications
- C. Memory storage: HIMEM, LOMEM, and variables

### Session IV

- A. The catalog and VTOC
- B. Reading and writing files
  1. Sequential
  2. Random access

Let's have your suggestions on course content.

## DATAWARE

Gourmet Food for Your Number Cruncher

The advent of microprocessing has placed sophisticated data processing equipment within the reach of analytically-inclined persons such as yourself. Unfortunately, no one has thought to provide comparably sophisticated data for you to process. Until now, that is.

MicroData Services has developed and continues to develop a library of data in forms that can be processed directly by the major micros. Many diskettes have been created, and more are on the way:

- Short Term Economic Indicators,
- Long Term Economic Growth Data,
- Transportation Networks--U.S. & Local,
- 1970 and 1980 Census Summaries, and
- Local and National Weather Data.

Washington Apple Pi members are invited to sample our data base by ordering an Introductory Disk containing 39 of the 300 time series in our Short Term Economic Indicators library. This disk contains:

- Manufacturers' New Orders (1/48-4/81),
- Change in Inventories (2/48-3/81),
- Personal Income (1/47-4/81),
- Standard & Poor's 500 Stocks (1/45-5/81),
- Index of Leading Indicators (1/48-4/81),
- Consumer Price Index (1/45-4/81),

and many more--over 16,000 observations in all, worth at least \$150 from the time sharing services. As with all data supplied by MicroData Services, our Introductory Disk comes equipped with thorough user documentation, including instructions on how to access the data from your BASIC programs using standard DOS commands--over 200 pages of information that describes all of the time series in the Short Term Economic Indicators library, and their use on your micro. On the disk, you will find a handy BASIC program that demonstrates the use of the Short Term Economic Indicators, and, of course, the 39 time series. Price: \$25.

To order, or for further information:

## MicroData Services

2813 64th Avenue  
Cheverly, MD 20785  
(301) 341-7412 Evenings & weekends



# PRESIDENT'S CORNER

## David Morganstein

This month I've got three things on my mind: the East Coast computer show to be held in September; Applesoft compilers; and, disk protection systems. Although a diverse collection, all connected to our Apples.

Before getting to those subjects though, I want to express a concern that I have. A recent conversation with a fairly active member got my attention and raised a flag of warning. The Pi has a clear objective to help new owners become acquainted with and get the most out of their Apple. The comments of this member, a person who after only a year of ownership is doing many of the things he thought only alchemists capable of, is beginning to ask himself, "what can I now get out of being a contributing member?" This is no idle question and it is one I want to consider.

There are many very knowledgeable and capable enthusiasts who have given many hours of time to help the Pi in a myriad of ways. What is it that keeps these folks going and what is it that causes others to shy away? Obviously, not every one has the same amount of time to help. However, as the former Librarian, I know that month after month of the same routine, feeling the same responsibility, trying to contribute on a volunteer basis, can get to you. Some of us just like to share what we know and teach others when possible. I am convinced that there is an equation of sorts, what you put in will be balanced by what you get out. Sometimes that's a kind of 'good feeling', but that alone will not keep inspiring everyone.

I return to the member whose comments led to this dialog. I hope to be sure that the level of material in the newsletter, meeting topics and SIG opportunities provides a chance for everyone, even the more experienced, to learn from others. We can not afford to lose the talents of the more experienced among us because we failed to recognize their need to learn as well.

The annual east coast computer show is scheduled for the 25-27 of September at the D.C. Armory, as last year. This is an interesting event which I enjoyed and am looking forward to attending again. Bernie Benson (736-0912) has offered to coordinate the volunteers who will be needed to man a booth throughout the weekend. All Pi members who assist will gain admission to the show. Last year we uncovered many new members through our ably staffed booth. Participation can provide revenues through library sales as well as swell our membership ranks. Please consider volunteering for a time

slot. Another way in which you can help is to offer any stimulating graphics displays you may have. We will have at least one machine there. Interesting programs can provide entertainment and demonstrate the Apple's skills.

The 'hottest' new software for the Apple may well be the raft of compilers for Applesoft basic programs. These packages convert already working basic programs into faster executing machine language programs. Two compilers are currently available, by On Line Systems and by Hayden, and one is being tested (by Microsoft). If you have several slow-going Applesoft routines you use often, you may find this development of interest. The price paid, however, may be worth comparing. First, the machine language versions are much larger than the basic versions. Second, you can not dynamically allocate array space (e.g. you can't say DIM A(N), where N is a variable). If your program is too large, you may have to break it into pieces to make it fit the compilers limitations.

The topic of software protection systems has found its way into many an editorial lately. It seems appropriate to include the Pi magazine in that set. Most of the editorials have taken a strong pro-protection stand, arguing for the rights of the software producer to protect themselves from financial ruin at the hands of unscrupulous software pirates.

I recently became acquainted with a consumers viewpoint that has not been expressed. I purchased two programs, one from Sub-logic and another from Broderbund. Both of these software firms are noted for high quality in their products and documentation. However, on trying to boot up their disks on my Apple II with language card, I found that the programs would not go. Both of the packages require Applesoft which I had pre-loaded onto the language card using the Basics disk. Unfortunately, the particular protection procedures used dumped the language on the card rendering the programs useless to me. Now I must say that both companies were very gracious when notified of this situation. They both indicated that they did not know this was the case since the protection systems were purchased from other companies. It is a formidable task to try your program out on all possible hardware configurations (as well I know from our Library efforts!!). However, had the disks not been 'locked up tighter than a drum', there would have been no problem using the programs. As it is I must return them since they are unusable on my machine. I'm not sure where this protection stuff will go. The Hayden compiler I mentioned comes with a special chip to be plugged into the game i/o connector, without the chip the program will not run. I certainly hope the consumers interests are not forgotten in the efforts of the protectors to 'protect'.

# There's only one place to buy apples.

AT FREDERICK COMPUTER PRODUCTS !!!

As an Authorized Apple Dealer we offer a full selection of both Apple Hardware and Software at low, low prices.

Also, as an Authorized Service Center we provide fast and efficient service on all Apple products.



## **FREDERICK COMPUTER PRODUCTS, INC.**

Microcomputer Systems And Peripherals

5726 INDUSTRY LANE-FREDERICK, MD.- (301)694-8884

 **apple computer**  
® Authorized Dealer

# QUESTIONS, QUESTIONS, QUESTIONS by

Mark L. Crosby

Q. Where can one get a copy of NIFFUM?

A. NIFFUM is on the Washington Apple Pi Disk #22. For those who do not know what this utility does, it converts from DOS 3.3 TO 3.2 format.

Q. What do the disk rings do?

A. Disk rings or hubs help to protect the center hole from damage and help to seat the disk properly in the drive. They contribute to longer disk life and allow the read/write head to better track the disk. Some manufacturers have included these as standard on each diskette.

Q. Is it possible to get the mixed mode high resolution graphics on Page 2 to display text? I have Applesoft in RAM and can move graphics to Page 2 but the text has problems. From what I can see on the screen it appears Applesoft occupies this text space.

A. You are entirely correct. RAM Applesoft does occupy that space (Page 2 Text). In the Page 2 high resolution mode, text display (or mixed-screen - poke -16301,0) would otherwise be possible (with ROM Applesoft). I would recommend getting a high resolution text generator for your needs (see CPlot on WAP Disk #31 as an example). This would permit text directly on the high resolution screen as necessary. While the generator takes additional memory, the result is usually worth it. In your situation, it may be the only solution.

Q. In Jay Thal's article on a \$10 joystick (June 1981), where can I obtain the Jim Pak/Jameco Joystick with 100K linear pots? Also, is it possible to get directions for this joystick endeavor with better help than "mount according to the schematic"?

A. Most electronics stores carry the brand mentioned, although they may have to order it. Also try the computer stores since some of them carry discrete parts as well. Some stores are: Arlington Electronic Wholesalers 524-2412 and Arcade Electronics 256-4610. The article assumes that you know how to solder and follow a schematic. If you don't, it would be advisable to seek a person who does to assist you in building the joystick. Although assembly is not complex nor difficult, a wrong connection could damage your computer.

Q. Please describe what "RASTER GRAPHICS" is.

A. "RASTER" refers to the area in a cathode ray tube upon which an electron beam "draws" an image. Television is an example of high resolution raster graphics. Often it refers to the

horizontal scanning lines formed by the beam. These lines are spaced closely together but can be easily seen close-up. The image is re-drawn 30 times per second in a raster graphics system. This requires external memory storage of the image, i.e., RAM. In a Direct View Storage Tube (DVST) system, the screen itself is the memory. In such a system, lines are drawn only once and then persist on the face of the CRT.

Q. We have a Hayes Micromodem and are having trouble signing on to the Washington Apple Pi bulletin board system. After we get connected we get garbage coming down the line and have to hang up. What is the parity and duplex required to sign on?

A. The WAP ABBS requires full-duplex with no parity. If that doesn't fix things, try another ABBS to be certain your modem is fully operational.

Q. I want to control the high bit on my Epson MX-80 parallel interface via the annunciators in the Game I/O connector (in order to access MX-80 graphics). What are the correct electrical connections? I have tried several set-ups and can't get it to work.

A. Unfortunately, adding this nice feature will void your warantee - and you will only get TRS-80 block graphics - not dot-addressable graphics. The modification information is available from: Epson User's Group, 1017 Trollingwood Lane, Raleigh, NC 27609. Include a self-addressed stamped envelope for a reply. You may be interested in GRAFTRAX, a ROM which can be inserted in the MX-80 that gives software-controlled, dot-addressable graphics. It lists for \$95.

Q. The DOS Toolkit Applesoft Programmers Aid "&X" function for variable cross reference crashes if the program has DATA statements. Any ideas?

A. I tried this and had no trouble cross-referencing a program with DATA statements. Perhaps there is something in your program the cross-reference can't handle. Have you tried running the program yet? If there were some sort of program error (incorrect format, garbage from an I/O error, etc.) it might crash as you described. Try loading (but do not run) your program from a cold start, i.e., from a boot of the Toolkit disk then try the cross reference.

Q. Do you know of an efficient way of loading Applesoft Program Modules from within an Applesoft Program without clearing the main program? I am looking for a way of loading main programs from a menu-loading subroutine module, running it, and returning to the menu.

A. I have worked with a system that uses CHAIN which functions similarly. The first program has all the DIMension,



# DISCOUNT HOUSE takes a BIG bite out of game prices.

PROGRAM	SYSTEM	MFG	LIST \$	SALE \$	PROGRAM	SYSTEM	MFG	LIST \$	SALE \$
Galactic Empire I	D48+	BS	24.95	19.95	Cartels & Cutthroats	D48M	ST	39.95	33.95
Galactic Triad II	D48+	BS	24.95	19.95	Torpedo Fire	D48M	ST	59.95	48.95
Galactic Revolution III	D48+	BS	24.95	19.95	Warp Factor	D48M	ST	39.95	33.95
Galactic I, II & III			74.85	55.00	Computer Quarterback	D48M	ST	39.95	33.95
Tawala's Redoubt IV	D48+	BS	29.95	22.95	Computer Conflict	D48M	ST	39.95	33.95
Galactic I, II, III & IV			104.80	75.95	Computer Air Combat	D48M	ST	59.95	49.95
Adventure	D48+	RC	29.95	25.95	Computer Napoleonic	D48M	ST	59.95	49.95
Stellar Trek	D48+	RC	24.95	19.95	Computer Ambush	D48M	ST	59.95	49.95
Datestones of Ryn	D48+	AS	19.95	15.95	Computer Bismarck	D48M	ST	59.95	49.95
Morloc's Tower	D48+	AS	19.95	15.95	Gorgon	D48M	SS	39.95	33.95
Temple of Apshai	D48+	AS	39.95	32.95	Space Eggs	D48M	SS	29.95	22.95
Hellfire Warrior	D48+	AS	39.95	32.95	Pulsar II	D48M	SS	29.95	22.95
Zork	D48+	PS	39.95	32.95	Autobahn	D48M	SS	29.95	22.95
<b>Special Sale</b>					Orbitron	D48M	SS	29.95	22.95
Ultima	D48+	CP	39.95	29.95	Gamma Goblins	D48M	SS	29.95	22.95
<i>Coming Soon — Ultima II — Revenge of the Enchantress</i>					Star Cruiser	D48M	SS	24.95	18.75
Hi-Res Adventures:					Cyber Strike	D48M	SS	39.95	33.95
#0-Mission Asteroid	D48+	OL	19.95	15.95	Sneakers	D48M	SS	29.95	22.95
#1-Mystery House	D48+	OL	24.95	18.75	Both Barrels	D48M	SS	24.95	18.75
#2-Wizard & Princess	D48+	OL	32.95	27.95	Phantom's 5	D48M	SS	29.95	22.95
#3-Cranston Manor	D48+	OL	34.95	28.95	EZ Draw 3.3	D48M	SS	49.95	39.95
Apple Oids	D48M	CP	29.95	22.95	<b>Special Sale</b>				
Akalabeth	D48M	CP	34.95	28.95	Pool 1.5	D48M	ID	34.95	26.95
Raster Blaster	D48M	BC	29.95	22.95	Shuffleboard	D48M	ID	29.95	22.95
3-D Graphics System	D48M	BC	39.95	33.95	Missile Defense	D48M	OL	29.95	22.95
Space Album	D48M	BC	39.95	33.95	Hires Soccer	D48M	OL	29.95	22.95
Trilogy of Games	D48M	BC	29.95	22.95	Hires Football	D48M	OL	39.95	33.95
Fender Bender	D48M	BC	24.95	19.95	Hires Cribbage	D48M	OL	24.95	18.75
Cosmos Mission	D48M	BC	24.95	19.95	Sabotage	D48M	OL	24.95	18.75
Adventure 1,2 & 3	D48M	AI	39.95	33.95	Paddle Graphics	D48M	OL	39.95	33.95
Adventure 4, 5 & 6	D48M	AI	39.95	33.95	Tablet Graphics	D48M	OL	49.95	39.95
Adventure 7, 8 & 9	D48M	AI	39.95	33.95	Hyper Head-on	D32+	BS	24.95	18.75
Alien Rain	D48M	BS	24.95	20.95	Galaxy Wars	D32+	BS	24.95	18.75
Alien Typhoon	D48M	BS	24.95	20.95	Tank Command	D32+	BS	29.95	18.75
Snoggle	D48M	BS	24.95	20.95	Devil's Dungeon	D16I	RC	15.95	12.95
The Prisoner	D48M	EW	29.95	24.95	Golden Mountain	C32+	BS	19.95	15.00
Operation Apocalypse	D48M	ST	59.95	48.95					

D = Disk	AI = Adventure International	ID = Innovative Design Software
C = Cassette	AS Automated Simulations	PS = Personal Software
48, etc. = Ks of Ram	BC = BudgeCo	RC = Rainbow Computing
+ = Applesoft	BS = Broderbund Software	SS = Sirius Software
I = Integer	CP = California Pacific	ST = Strategic Simulations
M = Machine code	EW = Edu-Ware	OL = On-Line Systems

- VISA, MC add 3% (include card number and expiration date).
- Money order, certified check, cashier's check or bank wire deposit preferred.
- Allow 3 weeks for personal check to clear.
- COD's require 10% deposit—all COD charges to be paid by customer.
- Include \$2.50 for postage and handling.
- DC residents add 6% tax.
- Prices subject to change without notice—all items subject to availability.

## DISCOUNT HOUSE

P.O. Box 40813  
 Washington, DC 20016  
 (202) 364-0273

Apple and Applesoft are registered trademarks of Apple Computers, Inc.

DATA statements and variables that will be used by the entire system. It then chains to a menu which CHAINS to other programs that all CHAIN back to the menu. It is not exactly what you might call "efficient" since the chaining process is relatively slow for programs requiring lots of variable space. It does, however, allow smaller program size since no variables need be dimensioned within. If programs are to be substantially different, i.e., have few variables in common, don't use CHAIN. Rather, use "RUN" from the menu. Then write each program to include an exit that runs the menu at the conclusion. A statement such as: PRINT CHR\$(4) "RUN MENU" at the end of each should be sufficient.

Q. How do I split a BASIC program across the high resolution areas? I have a large program that uses both hi-res pages and would like to increase the size of the program but I am up against the hi-res areas as it is.

A. See the June 1980 issue of CALL A.P.P.L.E. page 19 "Applesoft Program Splitter" by William Reynolds III. The October issue has a follow-up article "Applesoft Program Splitter Modifications" by Charles Kluepfel page 45. It is beyond the scope of this column to include the machine language programs here.

Q. What is the difference between BOOT13 and the DOS 3.3 BASICS disk?

A. Although there are some differences between these two machine-language programs, they perform essentially the same function. The former is designed to be BRUN from a 3.3 environment while the BASICS disk is designed to be booted only. When either is invoked, the environment is changed to allow booting 3.2 disks. Another difference is that the BASICS disk also provides Language Card owners with the other BASIC language at boot time.

Q. I have heard that a GOTO or GOSUB causes Applesoft to start back at the beginning of the program to search for the line number. Is there any way around this?

A. Yes. For all FORWARD GOTO's or GOSUB's the target line number must have a different high byte (and greater) than the current line number being executed. The test is done in hexadecimal, of course, so a line numbers 100 and 200 (\$0064 and \$00C8) have the same high byte whereas lines 200 and 300 (\$00C8 and \$012C) do not. Two sample programs below with execution times illustrate this.

```

1 REM          EXECUTION TIME
2 REM          32 SECS
3 REM
4 REM
5 REM
6 REM
7 REM
8 REM

```

```

9 REM
10 REM
11 REM
12 REM
13 REM
14 REM
15 REM
16 REM
17 REM
18 REM
19 REM
20 FOR I = 1 TO 10000
21 GOTO 22
22 NEXT I

```

```

1 REM          EXECUTION TIME
2 REM          20 SECS
3 REM
4 REM
5 REM
6 REM
7 REM
8 REM
9 REM
10 REM
11 REM
12 REM
13 REM
14 REM
15 REM
16 REM
17 REM
18 REM
19 REM

```

```

100 FOR I = 1 TO 10000
200 GOTO 300
300 NEXT I

```

Q. How can you prevent a program from being listed?

A. Try POKEing 2048 with a 1 at the beginning of your program. Then if the program is stopped and you try to list it see what happens!

ERRORS, ERRORS, ERRORS!!!

Last month, I published a short program that was supposed to show how to print lower case on a printer using an Applesoft program. Needless to say, it didn't work as planned. The program simply capitalizes the first letter of each word in A\$. Here is the correct program:

```

10 PRINT CHR$(4)"PR# 1"
20 A$ = " WASHINGTON APPLE PI"
30 PRINT A$
40 FOR I = 1 TO LEN (A$)
50 B$ = MID$(A$,I,1)
60 IF NOT UFLAG AND B$ < > " " THEN
   B$ = CHR$(ASC (B$) + 32)
70 PRINT B$;
80 UFLAG = 0: IF B$ = " " THEN
   UFLAG = 1
90 NEXT I
100 PRINT : PRINT CHR$(4)"PR# 0"

```

```

RUN
WASHINGTON APPLE PI
Washington Apple Pi

```



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

The newest entries in our Basic library are from the IAC. Numbered volumes 39 and 40, they include a contribution from Sydney Australia. Roger Keating, author of Operation Apocalypse has sent the IAC a disk of interesting programs from 'down under' (vol. 40). Also new in the library are two PIG disks, numbered 5 and 6. Hopefully the new PIG librarian, Dr. Ho, will review these for us.

Volume 39. Applesoft line writer - a programming aid for writing into a program for commonly used set up routines, such as DOS strings.

DPRING - used for dumping Visicalc files.

Error Handling - a program trapping errors, printing the line number and reason for the error.

General Ledger - what the name implies...

Skywriter - hires drawing routine, display title and allows user entered hires shapes.

Videotape Catalog - a mini dbms for tapes.

Volume 40. HGR Demo! - simple geometric shapes from trig functions.

Biorhythm - an old favorite

Target shoot - a text game

Lucy - formerly known as Eliza but with an expanded vocabulary

Talking calculator - a delightful integer math calculator with the voice of Roger Keating from Australia, coming thru your Apple speaker.

Mastermind - another old favorite

Commercially available software. Saturn Navigator (Sub-logic). This simulation was written by a member of the Jet Propulsion Lab. In hires, you can simulate the trip from Earth to Saturn. Several warnings are in order. First, you must have the Sub-logic A2-3D1 hires drawing package to use this program. Second, it will not work on an Apple II with language card. The disk protection system dumps the needed Applesoft. However, it is a marvelous simulation and worth trying. You begin by breaking out of Earth's orbit with sufficient force to cross the Saturn orbit. Next, you travel to Saturn making mid-course corrections along the way to assure a proper approach. These corrections are made by use of a hires approach view of Saturn. Upon arrival you attempt to go into orbit around Saturn, again aided by a view out the portal. If successful, you can next try to move into a close in orbit and dock with a space station there. Well done!!

Crush, Crumble and Chomp. (EPYX/Automated Simulations). From the makers of Temple of Apshai and related adventures a new perspective on hires games. Think of all the aliens destroyed by the thousands of computer and video game owners. Now is your chance to play the creature and seek revenge on the

## DISCOUNT HOUSE

takes a **BIG** bite out  
of computer costs.

PROGRAM/HARDWARE	SYSTEM	MFG	LIST \$	SALE \$
VISICALC 3.3	D48+	PS	199.95	169.95
VISIDEX	D48+	PS	199.95	169.95
VISIPILOT	D48M	PS	179.95	152.95
VISITREND/VISIPILOT	D48M	PS	259.95	220.95
VISITERM	D32M	PS	149.95	127.95
Desktop Plan II	D48M	PS	199.95	169.95
PFS: Personal Filing System	D48M	SP	95.00	76.95
PFS: Report	D48M	SP	95.00	76.95
Letter Perfect	D48M	LE	149.95	124.95
Edit 6502	---	LE	49.95	42.95
Lower case generator	---	LE	34.95	28.95
Epson MX 100 *New*	---	EA	995.00	855.00
Epson MX 80	---	EA	645.00	475.00
Epson MX 80 FT	---	EA	Ask about	
Epson MX 70	---	EA	our other	
MX 80 FT upgrade	---	EA	Epson prices.	
MX 80 Grafrax	---	EA	95.00	80.00
MX 80 X036 Cable (CPS)	---	EA	29.00	23.00
Z 80 Sottcard	---	EA	399.00	335.00
16K Ramcard	---	MS	195.00	159.95

D = Disk	EA = Epson America
48 = Ks of Ram	LE = LKJ Enterprises
+ = Applesoft	PS = Personal Software
M = Machine code	SP = Software Publishing
	MS = Microsoft

- VISA, MC add 3% (include card number and expiration date).
- Money order, certified check, cashier's check or bank wire deposit preferred.
- Allow 3 weeks for personal check to clear.
- COD's require 10% deposit — all COD charges to be paid by customer.
- Include \$2.50 for postage and handling.
- DC residents add 6% tax.
- Prices subject to change without notice — all items subject to availability.
- Hardware shipped via UPS FOB Washington, DC.

### DISCOUNT HOUSE

P.O. Box 40813  
Washington, DC 20016

(202) 364-0273

Apple and Applesoft are registered trademarks of Apple Computers, Inc.



# THE MTU HAT by Vance Giboney

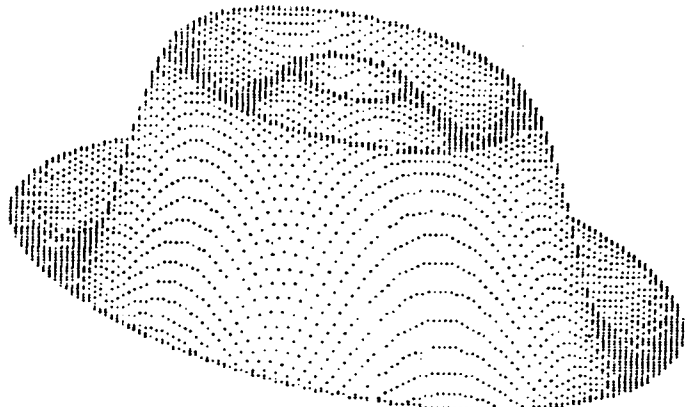
Earth. Yes, you select from a list of already created creatures (mechismo, goshilla, arachnis, kracken and others) or if you prefer, grow your own!!! Then you select a city with which to do battle (Washington, New York, Tokyo, etc.) Then the fun begins. As you move about the city grabbing and eating all that moves (people, police, national guard, tanks whatever), you can also obliterate, crush, burn or zap the buildings, parks and people. But beware the mad scientist flying about in his helicopter, he's out to get you. Sooner or later you'll be destroyed by the constantly attacking forces but you can take a lot of the city down with you. All in a delightful hires map which scrolls from one 5x5 set of squares to another withing a larger 30x30 map. One of the true high points of the game is the manual, it deserves a prize for cleverness and clarity.

VISIPLLOT (Personal Software). A very effective plotting package, the program's biggest advantage is that it links to VISICALC DIF files, allowing you to easily graph your visi-data. The package is otherwise not that superior to Appleplot. Some major differences are: you can build files containing several series of data points (where Appleplot allows only two series in one data file); the package comes with many graphics dump routines for a host of graphics printers (NEC, Paper Tiger and others) making it unnecessary to leave the program and use some external package. Several unusual but valuable display formats are available in addition to the scatter chart, bar chart and line chart possible with Appleplot. These include pie charts, area charts and hi/low displays.

Gamma Goblins (Sirius Software). This is not a Nasir game...how's that for an opening comment? The authors, Tony and Benny Ngo, have done a good job. Its just that when you hear Sirius Software, you expect something new and different. This is another invaders style game with you moving your ship left and right shooting up at the various invader shapes. You proceed from the lowest point value beasts (hyperdermics worth 20 pints) thru viroids (worth 40 pints), crash carts and corpuscles (50 & 60 pints) to the lymphoids (80). If you destroy enough of each you can finally make it to the nurses station and collect 400 pints. eh...the worst part is the manual which is mostly cutsy and not very informative. Of course, how much can you say about another invaders game?

In recent issues of Creative Computing, Micro Technology Unlimited has been running an ad that shows a PET computer with a rather nice version of the "cowboy hat" on its screen. The ad informs us that this hat was drawn using MTU's graphics package and the short 22-line program that they list down in the corner. Of course, anything that can be drawn on the PET can be drawn on the Apple as well--right? Naturally! Only MTU's program contains some funny-looking commands like WRPIX and GMODE (WRPIX?), the screen is 320 x 200 instead of 280 x 192, and even when this is taken into account, the hat comes out upside down on the Apple screen. Nonetheless, with a few well-placed changes in the program, MTU's hat can indeed be drawn with an Apple--thus saving us all the expense of a PET computer and MTU's \$495 graphics package.

```
]
10 HGR2 : HCOLOR=3: CLEAR
20 P=140: Q=100
30 XP=108: XR=1.5 * 3.1415927
40 YP=42: YR=1: ZP=48
50 XF=XR/XP: YF=YR/YP: ZF=XR/ZP
60 FOR ZI= -Q TO Q-1
70 IF ZI < -ZP OR ZI > ZP GOTO 150
80 ZT=ZI * XP/ZP: ZZ=ZI
90 XL=INT (.5 + SQR (XP * XP-ZT * ZT))
100 FOR XI= -XL TO XL
110 XT=SQR (XI * XI + ZT * ZT) *
XF:XX=XI
120 YY=(SIN (XT) + .4 * SIN (3 * XT)) *
YF
130 GOSUB 170
140 NEXT XI
150 NEXT ZI
160 END
170 X1= -XX + ZZ + P
180 Y1= -YY + ZZ + Q
190 HCOLOR=2: HPLLOT X1,Y1
200 IF Y1=0 GOTO 220
210 HCOLOR=0: HPLLOT X1,Y1 + 1 TO X1,191
220 RETURN
```



# COUNT SECTORS FROM APPLESOFT

William Schultheis

It can be very inconvenient to run out of space on a disk in the middle of a program. The following question appeared in a recent issue of Washington Apple Pi:

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 in a subroutine within an Applesoft program.

The solution turns out to be quite simple if you know something about the internals of DOS and a few tricks of assembly language programming.

DOS keeps a list of free sectors in the VTOC, which is stored on track 17, sector 0 of every standard disk. The VTOC provides information on the disk format so that DOS can read disks having different numbers of tracks and sectors. The VTOC includes a bit map which indicates the status of every sector on the disk with a bit which is zero if the sector is in use and one if the sector is free. A program can find out how the number of free sectors by scanning the bit map and counting the number of '1' bits. It would be possible to write such a routine in Applesoft, but it would take quite a few seconds to execute and would slow down any program that needed to use it very often. An assembly language routine would be much faster and take up less memory space.

In my opinion, one of the main reasons for owning any personal computer, and the Apple in particular, is that you are free to mess around in the system internals. You are missing half the fun of owning a personal computer if you use it like a commercial time-sharing system. To really get into the internals of a system, you need to learn the language it is written in. Both the monitor program and DOS are written in assembly language. So if you want your Apple to do something that the standard version doesn't do, it helps enormously to know how to produce an assembly routine which does it!

The difficulty of assembly programming tends to be over-rated. It does, however, require a more careful, systematic approach than Basic. So I thought it might be interesting for the novice or prospective assembly programmer to see how you go about designing an assembly language subroutine that does something really useful.

A programmer can save a lot of time and trouble by deciding first exactly what the program is supposed to do. But, you say, isn't the job of counting sectors so trivial that we can start coding right away? NOT SO! Suppose your system has two disk drives. Then you have to tell the subroutine which disk to use and maybe even which file you are writing. We can simplify the task a lot by saying that the routine will count the free sectors on the most recently accessed disk. That makes a lot of sense, because presumably you are interested in the amount of space on the disk because you have just opened a file or written to it. Not only is this assumption reasonable, it saves half of our work. Whenever DOS accesses a disk, it reads the VTOC into a buffer where it is used until you read or write a different disk. Since our program is going to work like an extension to DOS, it makes sense to use this buffer instead of reading another copy of the VTOC.

Until recently, assembly language programmers had no good source of information about the internals of DOS. The new book, Beneath Apple DOS by Don Worth and Pieter Lechner has changed that. DOS is full of all sorts of goodies for assembly hackers, and this book tells us where to find them. DOS has a VTOC buffer, and it lives at location B3BB in 48K Apples. (All locations are \$4000 bytes lower in 32K machines.) Within the VTOC, the bit map starts at \$B3F3 and ends at \$B47E. So our program has the almost trivial task of scanning a range of memory and counting bits.

This procedure will give us the right answer unless there is an open output file. When DOS is writing a file, it sets aside one track at a time, zeros out its bits in the VTOC, and creates a one-track bit map in the file buffer. Finding the buffer and the bit map is a complication we can defer until later.

One problem in writing assembly language routines to be used from Applesoft is knowing what memory locations can be used without clobbering Applesoft variables and internal data. A good location for small routines is the page starting at \$300. We will also need some page zero locations. This can be tricky, because Applesoft uses a lot of page zero. Fortunately, Applesoft does not use the locations reserved for the monitor. An assembly language routine can use many of the page zero monitor routines safely. The trick is to look at the monitor code and see how the monitor uses these locations. A routine which does a monitor type function can safely use the same locations which the monitor uses for similar functions.

The monitor has a useful service routine for scanning a sequence of memory locations. The routine is called NXTA1, and it is located at FCBA in the monitor. This routine is roughly equivalent to a FOR .. NEXT routine in BASIC. The routine is initialized by storing a starting address in A1L and A1H (\$3C and \$3D) and an ending address at A2L and A2H (\$3E and \$3F). Each time you call NXTA1, it adds 1 to the two-byte address in A1L,H. It then compares the new address to the address in A2L,H. If the first location is less than or equal to the second, the carry flag is cleared. When A1L,H is larger than A2L,H, the carry flag is set. A loop to examine a range of memory is coded as follows:

```
(INITIALIZE A1L,A1H,A2L,A2H)
LDY #0
LOOP LDA (A1L),Y
...
...
JSR NXTA1
BCC LOOP
```

Now is a good time to code all memory references as register equates. These equates define the interface between our routine, the monitor and DOS.

```
ORG $300 ORIGIN
A1L EQU $3C SCAN POINTER
A1H EQU $3D
A2L EQU $3E END OF SCAN
A2H EQU $3F
A4L EQU $42 SECTOR COUNT
A4H EQU $43
```

```
NXTA1 EQU $FCBA LOOP ROUTINE
MAPST EQU $B3F3 VTOC BIT MAP
MAPEN EQU $B47E END OF MAP
```

Before I start coding in assembly language, (or Basic for that matter) I like to sketch an outline of the program in pseudo-code. This is a sort of "Pison Pascal" which serves to define the logic of the program. For this routine it is quite simple:

```
INITIALIZE
LOOP
    PROCESS ONE BYTE
    POINT TO NEXT BYTE
UNTIL END OF BIT MAP
RETURN
```

Now we are ready to design a subroutine which counts the number of '1' bits in a byte. The strategy is to shift the byte left one bit at a time. After each shift, we check the carry flag. If it is set, the high order bit was a one and we add one to the sector counter. When the remainder after shifting is zero, there are no more '1' bits to count, and we can return. This trick works because the shift left instruction moves '0' bits into the right end of the word. After you shift a word left eight times, it is guaranteed to be zero no matter what you start with.

```
BYTE IN REG-A
LOOP
    SHIFT LEFT
    IF CARRY IS SET
        ADD ONE TO COUNT
UNTIL RESULT ZERO
RETURN
```

One trick is involved in coding the routine. Assume that the carry is set and we update the sector counter. The process of updating the counter clobbers the zero flag in the status word. The fix for this problem is to push the status word onto the stack before adding to the counter. We can now proceed to code the subroutine. But wait, why do we start at the bottom when everyone talks about top-down programming? The answer is that we have already written pseudo-code top down and know how the whole routine will fit together. Coding bottom up is usually best after a



detailed top-down design. That is because the higher level routines have to supply data to the lower level routines, and it is more efficient to write the calling sequence if you know exactly how the lower level routines work. Here is our subroutine:

```
*ROUTINE TO COUNT '1' BITS
*IN BYTE PASSED IN A-REG
BITS ASL A CHECK LEFT BIT
      BCC B2 BRANCH IF '0'
      PHP SAVE STATUS
      INC A4L TWO BYTE ADD
      BNE B1
      INC A4H
B1    PLS RESTORE STATUS
B2    BNE BITS LOOP UNTIL ZERO
      RTS
```

Now that we know how BITS works, we can code the main driver. It works just like the pseudo-code version except that we want to test each byte before calling BITS. If the byte is zero, there is no need to count bits. We will use the pre-defined equates, so there is no need to remember the addresses of things:

```
*MAIN SECTOR COUNT ROUTINE
LOOP LDA (A1L),Y GET BYTE
      BEQ L1 SKIP IF ZERO
      JSR BITS COUNT BITS
L1    JSR NXTA1 NEXT BYTE
      BCC LOOP TEST FOR END
      RTS RETURN TO BASIC
```

The last part of the program is the initialization of the sector count and the bit map start and end:

```
*INITIALIZE
      LDA #<MAPST
      STA A1L
      LDA #>MAPST
      STA A1H
      LDA #<MAPEN
      STA A2L
      LDA #>MAPEN
      STA A2H
      LDY #0
      STY A4L
      STY A4H
```

Now we can figure out how to track down the one-track bit map which will be lurking in the file buffer of any file

## STOCK OPTION ANALYSIS PROGRAM

Uses the Black-Scholes model to calculate the fair price of options. Then calculates expected profit (or loss) from any trade or spread.

Begins by calculating the prices of the stock over +/-3 std. dev. For each stock price the option price is calculated and the profit (or loss) determined. Each individual "transaction" is multiplied by its probability and summed to give the statistically expected profit (or loss). Can analyze Calendar (Horizontal) Spreads, Vertical Spreads, Straddles, involved spreads such as Butterfly Spreads, as well as Single transactions.

Most information can be stored on disk: an update program is provided. Files contain price and dividend information, expiration dates, and commission schedules. Commission schedules can be modified or new ones created.

Can be used with one or two disk drives and a printer. Hi-Res graphics can be saved on the disk for subsequent printing. A DEMO routine demonstrates these programs.

STOCK OPTION ANALYSIS PROGRAM \$250 Manual (alone) \$25

Req. Apple II/III+, DOS 3.3 and (48K RAM, ROM Applesoft, AUTOSTART ROM) or (Lang. Card).  
(Apple II and Applesoft are trademarks of Apple Computer, Inc.)

H&H Scientific specializes in investment software for the Apple II. Other products include STOCK MARKET UTILITY PROGRAMS and ELECTRONIC STOCK PACKAGE.

## H & H SCIENTIFIC

13507 Pendleton Street  
Fort Washington, MD 20744  
Tel (301) 292-3100



you are writing. The DOS buffers are stored just above HIMEM. Each buffer is 595 bytes long. It contains five main parts: a data buffer, a track/sector buffer, a file manager work area, a file name buffer, and a set of link pointers. The bit map we are looking for begins 33 bytes from the beginning of the file manager work area.

The address of the first buffer is stored at the very beginning of DOS, location \$9D00 in a 48K Apple. This address points at the file name buffer. If the buffer is in use, the file name buffer contains the 30 character name in ASCII code with the high bit set on. If the buffer is not in use, the first byte of the name buffer is set to zero. The address of the next buffer in the list is stored 36 bytes above the start of the name buffer. If the current buffer is the last one, the link address is set to zero. Another link vector, 30 bytes above the file name, points to the file manager work area.

We can find our bit map by searching the buffers starting with the one pointed to by location \$9D00 and following the link pointers. At each buffer we check the first byte of the name. If the byte is not zero, we know the buffer is in use. We take the file manager buffer pointer and use it to find the bit map. Then we load each of two bytes and call BITS to count the available sectors. We stop when we reach a buffer with a link pointer of zero. Here is the pseudo-code:

```

GET FIRST BUFFER
LOOP
  GET FIRST BYTE OF NAME
  IF NOT ZERO
    GET FILE MGR BUFF ADDR
    GET FIRST MAP BYTE
    COUNT BITS
    GET SECOND BYTE
    COUNT BITS
  GET NEXT BUFFER
UNTIL ZERO FLAG
RETURN

```

The search of the file buffers can be simplified by using three handy DOS subroutines. A routine at \$A792 stores the address of the first buffer at \$40,\$41 in page zero. A routine at \$A79A

updates \$40,\$41 to point to the next buffer. It sets the zero flag if the current buffer is the last. A routine at \$A7AA loads the first byte of the file name, setting the zero flag if the buffer is not in use. If you want to see how these routines work, use your Apple monitor to disassemble the code. Just CALL -151 and enter A792L. (It is VERY dangerous to call undocumented routines in DOS without knowing exactly what they do. Otherwise you could accidentally destroy DOS and the contents of a couple of disks!)

The complete program is shown in Figure 1. You can enter it and assemble it with the TED II+ editor and assembler found in Volume 8 of the Washington Apple Pi library. Sooner or later the assembled routine will show up in the library on a new utility disk. If you need the routine right away, the memory dump is listed in Figure 2, so you can enter it with the monitor. The program works with DOS versions 3.2.1 and 3.3 on a 48K machine. If you have a 32K Apple, change the register equates as follows:

```

MAPST EQU $73F3
MAPEN EQU $747E
PTBUF EQU $6792
USBUF EQU $76AA
NXBUF EQU $679A

```

In order to use the routine, BLOAD it at location \$300. You access it from BASIC by executing CALL 768. The routine leaves the sector count in locations 66 and 67. The complete sequence would be:

```
CALL 768: N=PEEK(66)+256*PEEK(67)
```

If you try this from a program and write out the result, you will see that the result is nearly instantaneous. As a final check, run FID to see if the two results agree.

```

1  *ROUTINE TO COUNT FREE SECTORS
2  *
3  *DEFINE LINKAGE
4  *
5          ORG $300
6  A1L    EQU $3C    SCAN POINTER
7  A1H    EQU $3D
8  A2L    EQU $3E    END OF SCAN
9  A2H    EQU $3F
10 A4L    EQU $42    SECTOR COUNT
11 A4H    EQU $43
12 NXTA1  EQU $FCBA  LOOP ROUTINE
13 MAPST  EQU $B3F3  VTOC BIT MAP
14 MAPEN  EQU $B47E  END OF MAP
15 PTBUF  EQU $A792  FIRST BUFFER
16 USBUF  EQU $A7AA  BUFFER IN USE?
17 NXBUF  EQU $A79A  NEXT BUFFER
18  *

```

```

19 *INITIALIZE
20 *
0300: A9 F3 21 LDA #MAPST
0302: 85 3C 22 STA A1L
0304: A9 B3 23 LDA #MAPST
0306: 85 3D 24 STA A1H
0308: A9 7E 25 LDA #MAPEN
030A: 85 3E 26 STA A2L
030C: A9 B4 27 LDA #MAPEN
030E: 85 3F 28 STA A2H
0310: A0 00 29 LDY #0
0312: 84 42 30 STY A4L
0314: 84 43 31 STY A4H
32 *
33 *MAIN SECTOR COUNT ROUTINE
34 *
0316: B1 3C 35 LOOP LDA (A1L),Y GET BYTE
0318: F0 03 36 BEQ L1 SKIP IF ZERO
031A: 20 4D 03 37 JSR BITS COUNT BITS
031D: 20 BA FC 38 L1 JSR NXTA1 NEXT BYTE
0320: 90 F4 39 BCC LOOP TEST FOR END
0322: 20 26 03 40 JSR COPEN LOOK FOR OPEN FILE
0325: 60 41 RTS RETURN TO BASIC
42 *
43 *LOOK FOR OPEN FILE
44 *AND COUNT ITS SECTORS
45 *
0326: 20 92 A7 46 COPEN JSR PTBUF START OF LIST
0329: 20 AA A7 47 NAME JSR USBUF BUFFER IN USE?
032C: F0 19 48 BEQ NB ZERO IF NOT
032E: A0 1F 49 LDY #1F OFFSET TO ADDR
0330: B1 40 50 LDA ($40),Y OF FILE MGR
0332: AA 51 TAX #WORK AREA
0333: 88 52 DEY #WHICH GET
0334: B1 40 53 LDA ($40),Y
0336: 85 3C 54 STA A1L AND SAVE IN
0338: 86 3D 55 STX A1H A1L,H
033A: A0 21 56 LDY #21, OFFSET TO SECTOR MAP
033C: B1 3C 57 LDA (A1L),Y GET FIRST BYTE
033E: 20 4D 03 58 JSR BITS AND COUNT
0341: C8 59 INY
0342: B1 3C 60 LDA (A1L),Y AND SECOND BYTE
0344: 20 4D 03 61 JSR BITS AND COUNT
0347: 20 9A A7 62 NB JSR NXBUF NEXT BUFFER
034A: D0 DD 63 BNE NAME IF IT EXISTS
034C: 60 64 RTS #ELSE EXIT
65 *
66 *ROUTINE TO COUNT '1' BITS
67 *IN BYTE PASSED IN A-REG
68 *
034D: 0A 69 BITS ASL #CHECK LEFT BIT
034E: 90 08 70 BCC B2 BRANCH IF '0'
0350: 08 71 PHP #SAVE STATUS
0351: E6 42 72 INC A4L TWO BYTE ADD
0353: D0 02 73 BNE B1
0355: E6 43 74 INC A4H
0357: 28 75 B1 PLP #RESTORE STATUS
0358: D0 F3 76 B2 BNE BITS LOOP UNTIL ZERO
035A: 60 77 RTS

```

```

0300- A9 F3 85 3C A9 B3 85 3D
0308- A9 7E 85 3E A9 B4 85 3F
0310- A0 00 84 42 84 43 B1 3C
0318- F0 03 20 4D 03 20 BA FC
0320- 90 F4 20 26 03 60 20 92
0328- A7 20 AA A7 F0 19 A0 1F
0330- B1 40 AA 88 B1 40 85 3C
0338- 86 3D A0 21 B1 3C 20 4D
0340- 03 C8 B1 3C 20 4D 03 20
0348- 9A A7 D0 DD 60 0A 90 08
0350- 08 E6 42 D0 02 E6 43 28
0358- D0 F3 60

```

### COMPUTER ACE

P.O. BOX 728 / ADELPHI MD. 20783  
PHONE: (301) 587-7459

APPLE w/16K.....\$1059  
APPLE w/32K.....\$1104  
APPLE w/48K.....\$1149  
DISK w/CONT.....\$ 535  
DISK only.....\$ 465  
PASCAL SYSTEM.....\$ 415  
INTEGER CARD.....\$ 150  
APPLESOFT CARD.....\$ 150  
SILENTYPE PRINTER..\$ 515

--- END ASSEMBLY ---

TOTAL ERRORS: 0

91 BYTES GENERATED THIS ASSEMBLY



# WORD PROCESSING SOFTWARE FOR THE APPLE: A SYNOPSIS OF THE PUBLISHED REVIEWS

by Walton Francis

There are now over two dozen word processing software packages available for the APPLE II. This article summarizes the information scattered in a like number of publications as an information guide to those considering buying a word processor or upgrading their current system.

## THE SOFTWARE

Listed below is each word processor program and the reviews which cover it, plus brief comments either summarizing the reviews or calling your attention to some particular feature of the system. These comments are not intended to be balanced or in depth. Systems which I believe to be especially useful in "word crunching" (such as writing long articles) because particularly easy to use in editing are marked with an asterisk. These are not necessarily the best systems for complex formatting needs such as technical articles, for form letters and mailing lists, or for other specialized applications. Nor are these the most cost-effective systems for many users. I have, in addition, marked some of the better buys with an exclamation mark. Thus, all 80 column systems get an asterisk because they are considerably easier to use than 40 column systems, but only a few 80 column systems are inexpensive enough to be serious candidates for best buys. All of these judgments are partial and tentative. I have only used a few of these systems and some I check rate have not even been reviewed yet--with my rating based on reading between the lines in the ads, a risky approach.

(a1) Apple PIE (aka Word Processing System), \$130, Programma International Inc, 2908 N Naomi St, Burbank CA 91504 (800-423-2978). Reviewed in 6, 11, 18,

20, 21, and 23. PIE has received more top reviews than any other 40 column system (see, for example, sources 18 and 23). This reflects its sophistication and power. The price one pays is that PIE is apparently the most difficult to learn of the popular word processors.

\*(a2) Apple PIE 80 column versions, \$130 (plus 80 column card and CRT), same address as above. As yet no reviews of these new versions. Unfortunately, unlike most other 80 column systems, you must buy PIE configured to a particular board.

(b) Apple Writer, \$75, Apple Computer Inc, 10260 Bandley Drive, Cupertino CA 95014. Can be supplemented by Apple Writer Extended, for \$30, from Brillig Systems Inc., 10270 Fern Pool Court, Burke VA 22015 (703-323-1339), which adds many missing features and greatly expands ability to edit and manipulate text files (see review in 21). Reviewed in 2, 3, 11, 18, 20, 21, and 23. The most extensive review (18) concludes that this program is fast, direct, and easy to use, with a good manual, but hampered by lack of wrap-around and absence of more sophisticated formatting features. Another review (11) describes it as "functional....takes concentration and skill to operate...and...takes away from the creative and refining process". Written in machine language and unlocked, Apple Writer is easily modifiable (by programmers) to add features, such as the Paymar chip (3). It does not use all Centronics printer capabilities, and disk access is a nuisance.

(c) Copywriter +, \$395 (plus Z-80 card), Digital Marketing, 2670 Cherry

Lane, Walnut Creek CA 94596. This program is probably but not certainly runnable on APPLE CP/M. The review (8) summarizes it as "a powerful Pascal word processor and mail merge program....However, print control is through embedded commands in the text, as opposed to on-screen formatting" and "The need to move back and forth between edit and print programs is painful". Recommended only for commercial, high-volume mailers.

\*(d) EasyWriter, \$100, Information Unlimited Software, 281 Arlington Avenue, Berkeley CA 94707. Reviewed in 1, 4, 5, 6, 18, 20, 21, and 23. I do not believe that a 40 column system could be easier to learn and use than EasyWriter. Its main defects are that it lacks the formatting sophistication of such systems as PIE and Super-Text, and one cannot use the Paymar chip to get lower case on screen.

\*(e) EasyWriter Professional, \$250 (plus 80 column board plus CRT), address as above. Reviewed in 1, 8, 20, 21, and 23. With the 80 column card and a fully visual approach the Professional system is excellent. It is much cheaper than the CP/M systems but I can't rate it a best buy in view of the prices for PIE and Letter Perfect.

!(f) Electronic Typing Program, free if you're willing to spend a few hours typing the program in. This line oriented text editor is quite primitive, and won't save files, but includes a few sophisticated features such as text wraparound. The program is provided in 12 (you will need Integer Basic as well; the IAC version works). Try it--it works and is educational.

(g) Hebrew II, \$60, Aurora Systems, Inc., 2040 East Washington Avenue, Madison WI 53704 (608-249-5875). Mini review in 21.

\*(h) Letter Perfect, \$150 (plus Paymar chip and optional 80 column board and CRT), LJK Enterprises Inc, PO Box 10827, St Louis MO 63129 (314-846-2313). No reviews available yet. Ad indicates that system uses machine language, single load disk system, and mnemonic key codes, and

supports all printers. Because this system works in both 40 and 80 columns, it may be a super starter system while you ponder the further investment in the 80 column board.

\*(i) Magic Wand, \$495 (plus Z-80 card), Small Business Applications, 3229 Louisiana, Suite 205, Houston TX 77006. Reviewed in 8, 17, and 23. This is a serious system, second only to WordStar in popularity among CP/M users. Peachtree just acquired it to add to their product line. If only the price were lower, this might be the single best system.

(j) Magic Window, \$100, Artsci Inc, 10432 Burbank Blvd, North Hollywood CA 91601 (213-985-2922). Reviewed in 2, 10, 11, 18, 21, and 23. This system attempts to provide a fully visual approach (what you see is what you get) within the confines of the 40 column screen. When and if it is available in an 80 column version it may be one of the best; for now I would skip it as obsolescent.

(k) Master Text Processor, \$140, Charles Mann and Assoc., 7594 San Remo Trail, Yucca Valley CA 92284 (714-365-9718). Mini review in 21. An easy to use program with graphics capabilities and form letter and mailing list capabilities built in.

(l) Pascal, \$500 (80 column board and CRT optional), address same as Apple Writer. Applications suggestions in 13, 14, 19, and 25. One short article (17) says Pascal is "creditable" but that "it does not have a very good human interface" and the manual is not easy for beginners. All four applications articles cover generating upper and lower case characters, number 25 using a Paymar chip. Number 14 also discusses other aspects of Pascal as word processor. Since Pascal can use all 80 column boards, it can provide a full visual approach--what you see is what you get.

(m) Pro-Type, \$75 (plus Z-80 card), Interactive Microware, PO Box 771, State College PA 16801. No serious review available. One source says it will work with APPLE, but no confirmation. According to (20), a

line editor rather than true word processor, in assembly language, with good form letter capabilities.

(n) Scribe, \$80 (plus Paymar chip), Datacope, PO Drawer AA, Hillcrest Station, Little Rock AR 72205. Reviewed in 18 and 21. This is clearly a solid system, but is not apparently outstanding in any particular way.

(o) Select, \$?, Information Systems, 919 Sir Francis Drake Blvd, Kentfield CA 94904 (415-459-4003). This is a CP/M system now available for the APPLE. No reviews are available, but the ad claims it is very easy to learn and use.

(p) Spellbinder, \$495 (plus Z-80 card), Lexisoft Inc, Box 267, Davis CA 95616. Another CP/M system so far unreviewed.

!(q) SuperScribe (formerly SuperScript), \$90, On-Line Systems, 36575 Mudge Ranch Road, Coarsegold CA 93614 (209-683-6858). This new system is as yet unreviewed. The ad states that it uses machine language, gives upper and lower case without hardware, and works with any printer. Since you need not buy a Paymar chip to get lower case, this is probably a best buy in its class (Apple Writer, Scribe, Magic Window). It may even deserve an asterisk and beat out EasyWriter.

(r) Super-Text II, \$150, Muse Software, 330 N Charles St, Baltimore MD 21201 (301-659-7212). Reviewed in 1, 5, 11, 13, 18, 20, 21 and 23. Like PIE, this system is powerful but complex. Perhaps it deserves an asterisk--it certainly deserves consideration as one of the top 40 column systems.

(s) Text Editor Ver 3.0, \$60, Peripherals Unlimited, address unknown. Mini reviews in 21 and 23. Slow, line editor rather than true word processor. Separate mailing list and form letter packages can be purchased.

(t) Text Power (aka TXT/ED 2.0), \$80 (plus optional 80 column board and CRT), Systems Design Lab, 2612 Artesia

Blvd, Suite B, Prepondo Beach CA 90278 (213-374-4471). Reviews in 21 and 23. Powerful system but not a true word processor because line rather than character oriented.

!(u) The Correspondent, \$45, Southwestern Data Systems, PO Box 582-W, Santee CA 92072 (714-562-3670). Reviews in 6 and 18. Among the line oriented text editors, this is probably the most sophisticated and powerful. Probably the cheapest system worthy of consideration by a serious user.

\*(v) The Executive Secretary, \$250 (plus optional 80 column board and CRT), Personal Business Systems Inc, 4306 Upton Avenue South Minneapolis Minn 55410 (612-929-4120). Review in 2. This may be the best word processor for the APPLE II, bar none. Its list of features is unbelievable, including form letter and mailing list, mini data base manager, and Data Factory and VisiCalc interfaces built in. Its print formatter capabilities appear second to none. The review (2) states that "considering the options it has, it is amazingly simple" to use. Like Letter Perfect, one program works with both 40 columns and most 80 column boards. Tentatively, I rate this a best buy in spite of its price.

(w) TOUGH PLUS Apple Text Processor, \$25 (or free after typing a long program), Nibble, Box 325, Lincoln MA 01773 (617-259-9710). Described and listed in 15. Another cost-effective line editor, best compared to The Correspondent. It may be an equally good buy.

(x) Vedit, \$110 (plus Z-80 card), CompuView Products Inc, 618 Louise, Ann Arbor MI 48103. No available review, but this CP/M system may work with APPLE.

\*(y) Word Painter, price not available (requires 128K APPLE ///). Mini review in 21. Without question better than any word processor for the APPLE II, since it includes a fully visual editing approach, 80 columns, and a



professional keyboard far superior to the II's.

(z) Word Power, \$100 (plus optional 80 column board and CRT), Systems Design Lab, address as above under Text Power. This line editor reviewed in 21 and 23. Same as Text Power but includes mailing system.

\*(aa) Word Processor II, \$75 (plus optional 80 column board), Software Solutions, 2462 Westington, St. Louis MO 63043. Mini review in 21. Sophisticated system, including mailing list feature, specializing in editing text and other BASIC files. A serious review might show this system, which like Letter Perfect and The Executive Secretary works with all equipment configurations, to be a real winner.

(bb) Word Processing System I, \$70, CompuSoCo, 26251 Via Roble, P.O. Box 2325, Mission Viejo, CO 92690 (800-824-7888). Mini review in 21. A powerful editor.

(cc) Word Processing System II, \$140, address as above. Mini review in 21. Includes mailing list and form letter systems in addition to features in System I. Formatter allows printing of letters in graphics type format.

(dd) WordStar, \$375 (plus Z-80 card plus 80 column board plus CRT), MicroPro International, 1299 4th St, Suite 400, San Rafael CA 94901 (415-457-8990). Reviews in 9, 17, 20, and 23. The best selling micro computer word processor by far--"what you see (on the screen) is what you get". The only real problem with WordStar, aside from price, is that in actual use editing is not optimally easy--requiring, for example, an extra key stroke for some common commands.

(ee) WpDaisy, \$495 (plus Z-80 card), InfoSoft Systems Inc, 25 Sylvan Road South, Westport CT 06880. Reviewed in 16.

(ff) Write-On I, \$100, Rainbow Computing, 9719 Reseda Blvd, Northridge

CA 91324 (213-349-5560). Reviews in 1, 21, and 23. A line editor rather than true word processor. Several reviewers, nonetheless, rave about it, so it must be a good line editor.

(gg) Write-On II, \$150, address as above. Similar to I system, but especially sophisticated form letter and mailing list capabilities. Reviewed in 1 and 21.

### TRUST THE REVIEWS?

It might seem, to the casual eye, that one review is much like another in its attention to detail and conclusions. Unfortunately, nothing could be further from the truth. Consider, for example, comparative reviews of EasyWriter and Super-Text, two systems which probably account for very close to half of total sales (the other two big sellers are Apple Writer and Apple PIE--these four systems together have around 90 percent of the market). Of our sources, ten review EasyWriter (EW) or Super-Text (ST) or both. Many reach clear judgments directly bearing on our argument:

Review	EW Rating	ST Rating	No. Rated
(1)	unclear	unclear	five
(3)	not rev	rave	one
(4)	favor	not rev	one
(5)	even	even	two
(6)	dislike	not rev	one
(11)	not rev	first/tie	six
(13)	not rev	favor	one
(18)	fourth	sixth	seven
(20)	unclear	unclear	six
(23)	third/tie	third/tie	seven

Considering that most of these reviews are comparing many of the same systems, the lack of consistency is appalling. Super-Text, for example, ranges from a first place tie to sixth place. And the actual details of the reviews are sometimes enough to make one cry. Source 23, for example, actually rates a \$60 text editor (not a true word processor) ahead of EasyWriter on a price/performance basis, and rates Apple Writer as easier to use than EasyWriter. This conclusion is absurd,

considering that the main point of most EasyWriter reviews is ease of use--source 5, for example, states that "it is easy. There is just no comparison....If ever a program deserved the name it has, this one does".

This does not mean that one should despair at the reviews, but read them with care and compare them. Remember that authors in computer magazines tend to be entranced by sheer technical virtuosity--by exotic features of little or no relevance to you. Many of them, even those who try to explain word processing (see 2 and 23) really don't understand which features are of value in the real world. So study and compare with a grain of salt.

My second suggestion is to focus on those features of concern to YOU. Apple Writer may be good at handling text files for BASIC programmers, but if you plan to write the great American novel it is not even a minimally acceptable program compared to many of the others. The only features which are of near universal concern are ease of learning and ease of use (see my article in February 1981 Washington APPLE Pie). For the hundreds of additional features which one or another program has, concentrate on a key half-dozen or so and try to ignore the others, simply to avoid being overwhelmed by detail.

## RECOMMENDATIONS

I can't tell you which processor to buy, which is a real shame given the variety of choices and immense volume of literature facing you. What I can suggest is that you read at least a few articles (particularly those asterisked) to try and narrow your choice down. Then read some more articles on the particular programs which most attract you.

If all else fails, consider limiting yourself to the four bestsellers mentioned above. The great majority of users will not go wrong with any one of those programs, though I don't

recommend any of them as best buys. After all, something must have impelled the authors of Letter Perfect and SuperScribe to take on the entrenched sellers.

Don't forget the 80 column option. There's a reason why WordStar is the all time best selling word processor, and why a half-dozen 80 column systems have become available within the last few months. An 80 column system will cost you \$500 more in hardware, and perhaps several hundred dollars more for the program--but a lot of people believe that it is worth almost any price if "what you see is what you get". Perhaps the best choice of all is to buy one of the new 40 and 80 column compatible systems, such as Letter Perfect or The Executive Secretary, and keep your 80 column options open.

Finally, try out your tentative first choice before buying it. If you can't do simple editing after half an hour with the manual, don't buy it.

## THE REVIEWS

Listed below are each of the books or magazines and articles, and the software reviewed in each. The most useful reviews--generally those containing the most COMPARATIVE details-- are marked with an asterisk.

\* (1) BYTE, June 1981. "Four Word Processors for the Apple II by Keith Carlson and Steve Haber (Write-On, Super-Text, Scribe, EasyWriter, EasyWriter Professional).

(2) CREATIVE COMPUTING, July 1981. "The Executive Secretary" by Dale Archibald, "Fundamentals of Apple Writer" by Barry Bayer, "Through the Magic Window by Al Evans, "A Primer for Word Processing" by Gordon McComb.

(3) CREATIVE COMPUTING, February 1981. "Lower-Case Display for Apple Writer" by John Stith, and "Apple-Cart" by Chuck Carpenter (Super-Text).

(4) CREATIVE COMPUTING, October 1980.

"EasyWriter" by David Ahl.

(5) CREATIVE COMPUTING, July 1980.  
"Super-Text vs EasyWriter" by Voyle Glover.

(6) INFOWORLD, July 20 1981. "The Correspondent: A Screen/Cursor Text Editor" by Donald Teiser, "EasyWriter and EasyMailer from IUS" by David Ness and Alan Krigman.

(7) INFOWORLD, May 25 1981. "Apple Pie Word Processing System" by Donald Teiser.

(8) INFOWORLD, March 30 1981. "Copywriter +: Big Word Processor for Micros" by Victor Heyman, "EasyWriter Professional from IUS" by Thom Hogan, "Small Business Applications Magic Wand" by John Zussman.

(9) INFOWORLD, February 16 1981. "MicroPro International's WordStar" by Ed Martino.

(10) INFOWORLD, November 10 1980. "Magic Window WP System from Artsci" by Philip Tubb.

\*(11) INTERFACE AGE, May 1981, "Word Processing Apple-ications" by Robert Moskowitz (Apple PIE, Apple Writer, EasyWriter Professional, Magic Window, Super-Text, Write-On II).

(12) MICRO, July 1981, "Electronic Typing Program for the Apple" by Thomas Brock.

(13) NIBBLE, Number 4 1981. "Super-Text II Review" by Leslie Schmeltz, "Lower Case in Apple Pascal-at No Charge!" by James Florini.

(14) NIBBLE, Number 3 1981. "Pascal Text Processing" by Larry Litwin.

(15) NIBBLE EXPRESS, Vol 1. "T.O.U.G.H.--Text Processing System" by Mike Harvey.

(16) ONCOMPUTING, Fall 1980. "The New WpDaisy: Word-Processing Software" by Bob Magruder.

\*(17) ONCOMPUTING, Summer 1980. "A Writer Looks at Word Processors" by Jerry Pournelle, "Word Processors: A Look at Four Popular Programs" by Larry Press (Magic Wand, WordStar and other CP/M software), and "Using the Apple Pascal Text Editor as a Word Processor" by Jef Raskin.

\*(18) PEELINGS II, Nov-Dec 1980. "Special Word Processing Section" by various authors (Apple Writer, EasyWriter, Magic Window, Apple PIE, Scribe, Super-Text, Correspondent).

(19) PERSONAL COMPUTING, May 1981. "Generate Lower Case Characters with Pascal" by Sam Gaylord.

\*(20) PERSONAL COMPUTING, January 1981. "Word Processing Software Roundup" by Steven Jong (Apple Writer, EasyWriter, EasyWriter Professional, Apple PIE, WordStar, Super-Text, and several CP/M systems).

(21) SKARBK SOFTWARE DIRECTORY, 1981. (brief descriptions and mini reviews of 16 Apple word processors).

(22) SOFTALK, July 1981. "Marktalk Reviews" (Apple Writer Extended features).

\*(23) THE BOOK OF APPLE COMPUTER SOFTWARE, 1981, by James Sadlier and Jeffrey Stanton. "Word Processing" by the editors (introductory material, mini reviews of 9 Apple word processors including comparative ratings of 7), "CP/M Word Processors" by the editors (reviews and compares WordStar and Magic Wand, both CP/M systems, to EasyWriter Professional).

(24) WASHINGTON APPLE PI, February 1981, "The Search for the Almost Perfect, Low Cost Apple Word Processor" by Walton Francis.

(25) WASHINGTON APPLE PI, June 1980. "Blaise Away: Dan Paymar meets M. Pascal--Lower Case Input for your Pascal Apple" by Dr. Wo.



# THE SYSOP REPLIES

by John L. Moon

## INTRODUCTION

This article is a description of the Washington Apple Pi Apple Bulletin Board System. It describes what the system is, gives some background information as to how the system was developed, and instructs how you can begin to use the system. It will serve as a reference for new users of the WAP ABBS to give them enough information to set a signon for the WAP ABBS and begin using the system.

The WAP ABBS is an Applesoft program of about 1600 lines written by me on a continuing basis since last September. There are currently about 200 members of WAP that have signons to the system. On a typical day, over 40 people call in and log onto the system. Because of the length of the program, I have not included a listing with this article.

The WAP ABBS runs on an Apple II at my house; it requires a D.C. Hayes Micromodem and at least one disk in order to run. Either the Applesoft firmware card or the Language System is needed because of the large program and its equally large in memory indicies and other data storage (I use a Language System card). The program runs under DOS 3.3. The computer has its own phone line in our house.

## ABOUT THE PROGRAM

Because of the nature of this application, it must be very reliable—therefore, a number of steps have been taken to reduce the possibilities of system crashes. For example, all input is with a subroutine specially written for the WAP ABBS that uses GET so that CTL-C will not cause a break. CTL-D is ignored on input so that no DOS commands imbedded in messages will be unintentionally executed. All input line lengths are checked so that messages can not be overwritten unintentionally. The modem runs in transparent mode so that the normal default controls of the micromodem are disabled so it will not go into

terminal mode, reset my computer or other functions. The forsoins is not a challenge for people to find holes in the system security—the intent is for the system to be able to run unattended for several days at a time.

As the system operator, I have put a number of commands in the system that only I can invoke. Examples of these are commands that tell the system to operate in a "local" mode where I can sign on to the system from the computer keyboard and it will ignore the phone until I tell it to go "remote" again. I also have special commands to set the date (I wish I had a clock card...) and monitor or set certain other status information.

Several of the commands on the system essentially just print out a text file from the disk. Examples are the signon bulletin, the meetings file, the help file, and the file on other BBS systems. To update these files, I have to bring the WAP ABBS totally down and then update the file using a text editor. The WAP ABBS must be restarted after such updates. The system must also be brought down to update the WAP ABBS itself including adding new passwords since they are currently stored as DATA statements in the program. To restart the system takes about two minutes while it reinitializes the memory pointers to the messages.

For all its size, the WAP ABBS program is quite simple in structure. It is divided into sections that do the initialization, the call answering and logon, the command processing, and finally the call clearing. By far, the most code is in the command processing where there is a section of code for each possible command that can be entered. There is a set of common subroutines for disk and modem I/O as well as other commonly used functions.

The primary data structure is the message file on disk. It is a random access text file, 100 records of 452 bytes each. The first 40 characters of each record contain the information of from, to, date, and subject. The rest of the record can contain up to 10 lines of data, each up to 41 characters long (including the carriage return at the end of each line). During initialization, the first forty characters of each record is read

into an in memory character array. Most commands will normally use the information in memory to locate or summarize a message. When it is necessary to do disk I/O, the file is opened, updated and closed without any dependence on a user input (excepting upload). This opening and closing each time was found to be necessary to flush the output buffers. In the case of a hardware failure this ensures the integrity of the file.

## OPERATING HOURS

The WAP ABBS is typically up 24 hours a day, 7 days a week. The exceptions are:

1) When it crashes. I restart it whenever I notice that it has crashed which may be several minutes to several days after the crash.

2) Whenever I want to use my computer. I take the system down, do whatever I want (for example, writing this article!), then bring it back up when I am finished.

3) When the machine breaks. The only item that has broken so far has been a power supply and this only after 9 months of being on 24 hours a day; it took about 3 days to get that fixed.

Generally, I check in on the system daily.

## GETTING A SIGNON

To get onto the system, you must have access to a terminal with a Bell 103A compatible modem. The most common configuration is an Apple with D.C. Hayes Micromodem, but I have folks on the system with equipment ranging from TRS-80's to Lear Seisler terminals. Assuming you have the hardware, then:

1) Obtain a signon. The signon is composed of your WAP number, your last name, and a password. You can either set it from me at a club meeting or by calling the club phone as described in the

newsletter.

2) When you have your signon, then call the WAP ABBS in full-duplex mode, originate mode, using 8 bits with no parity, 1 stop bit, and 300 baud (this is the default if you are using a Micromodem).

3) The system will ask for a carriage return and then for you to enter your signon. You get two chances to set it right before it treats you to a message telling you to set a signon. The signon should be entered without any blanks exactly as given to you.

4) The system will then ask if you want the club bulletin. Normally you will so answer "Y". A "N" or just a carriage return will go on without printing the signon bulletin.

5) If you are a beginner, your first command should be "H" for help. Your next command should probably be "I" for information which describes all the functions of the system and you should remember to always signoff using the "G" command.

## Q'S AND A'S

The following are some of the common questions and answers related to the bulletin board.

(Q1) How can I print out information from the WAP ABBS on my printer?

Assuming a D.C. Hayes or Apple Comm Card, at any pause for command or even prior to dialing the system, do a ctl-A ctl-X which puts you back into Basic, then do the PR#slot for your printer, then ctl-A ctl-F to set back into terminal mode. Depending on your printer, you may have to use the "N" command to turn linefeeds or delays on after carriage returns.

(Q2) What does the "K" command do?

It turns off a lot of the prompts. Instead of seeing "COMMAND?" you would just see the "?". The "K" command tosses, so if you use it



again, you are back where you started.

(Q3) What happens if I get disconnected without doing a "G" command?

You should call back immediately—the system will just hang in the middle of your session where it was disconnected waiting for someone to call. Sometimes, you will dial in to the system and instead of getting the signon request, you will already be in the system; if so, then someone got disconnected or forgot to use the "G" command. In this case you should use the "G" command for them. You don't have to call back after it says "THANKS FOR CALLING WAP..." Just wait about 20 seconds and hit a carriage return to recycle back to the signon.

(Q5) I dialed in once, it answered but I got nothings... whatever I typed appeared but it never seemed to respond... what happened?

The system was probably accidentally put into "chat" mode. If this happens to you, then try typing the word "resume" as the first word on a line. If it is in chat mode, then this will put it back into normal mode. You will then have to sign someone off as described above.

(Q6) I can't put beeps into messages. Why?

It was waking me up in the middle of the night - so I changed the program to throw beeps into the bit bucket.

#### ADVERTISING RATES

The following table shows our new advertising rates. Our newsletter distribution is about 1200 copies now, with about 200 of these going around the country. If you would like to advertise please send camera ready ad copy (black and white only, no halftones) by the 10th of the month to our PO Box.

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

## New!! A Finance Data Base System

... use double entry  
without being an accountant

- Balance Sheet
- Budgeting
- Save at Tax Time  
with Detailed Records
- Efficient Storage/Retrieval
- Human Engineered

Comprehensive manual in  
handsome 3-ring binder,  
(Requires DOS 3.3 and 48K)

## The ACCOUNTANT FINANCE DATA BASE SYSTEM

Send check for \$89.95 or  
VISA/MC (#, EXP. date) to:  
DECISION SUPPORT SOFTWARE  
1438 Ironwood Drive  
McLean, VA 22101 (703)241-8316

### ORDERIDENT

#### A Multiple Entry Key to Orders of Insects

Is an interactive computer program to help students identify insects and stimulate interest in insect biology.

Can identify any North American insect to one of the twenty six orders in the Class Insecta.

Will lead the student step-by-step through a series of questions until an identification is obtained.

Permits repeated attempts, if needed, with no more than a 'yes/no' response required.

Is completely self-prompting with easy to understand instructions at key points.

Has full documentation including a sample run.

Was developed by a professional entomologist for the APPLE II and APPLE II PLUS computers.

Does not require a printer.

INTRODUCTORY OFFER: \$44.95 on diskette  
or cassette.

U.S. shipments add \$1.50 for shipping and handling.  
Outside U.S. add \$2.50 for shipping and handling.  
Virginia residents add 4% sales tax.

Modules under development to identify insects to Family

### EDUCATIONAL COMPUTING

3144 Valentino Court  
Oakton, Virginia 22124

# PUFF IN (DOS TO PASCAL TEXT FILE CONVERSION) by Dr. Wo

A couple of months ago some of my club mates who were introducing themselves to Pascal asked me about the possibility of swapping files between DOS 3.3 and Pascal. Given that both systems use 16 sector disks I replied "of course you can swap files!" One day I finally got 'round to proving it.

Being an inveterate, chauvanistic Pascal programmer the project turned out to be both a little harder and a little easier than I first estimated. Harder because I didn't know a whit about DOS, so I had to learn how DOS keeps track of files. And easier because I had (and still have) no desire to map Pascal files into DOS files so I dismissed that part of the project right off!

Of course only one half of a suite of programs to communicate between operating systems is less than useless unless you really are a chauvanist. Fortunately, one of my club mates, Dana Schwartz, came to my rescue and supplied the other half. His program appeared in the last Apple Pi issue.

Actually, I owe Dana more than his half. It was he who came up with the names for the programs. Besides the reference to MUFFIN, the names allude to the translation from DOS's High-level language (the huff) to Pascal (the puff) and back again... or so he says. Sounds like a bit of fluff to me.

Like any good idea, or at least any idea for software, there turns out to be several variations on these programs. For example, our Pascal SIG is rife with home brew translation programs which are now in our library. In addition, the Carolina Apple Corps has a library disk with such programs on it (\$10), and Lee Meador of Dallas has an article in the most recent issue of the Apple Orchard on the topic. Also, there are several commercial

versions, the BRIDGE (\$90!!) from an outfit in North Carolina, and another program from Gryfon software in Silver Springs Md. Undoubtedly there are more.

What Good is it?  
=====

Any set of programs which maps between operating systems is, of course, interesting just for what it tells you about differences between the systems. But such programs go beyond that by increasing the utility of both systems. The programmer can select the best from two worlds.

From my point of view, the payoff from these mapping programs could be substantial for the BASIC programmer (but then I don't know why anybody programs in BASIC anyway). Using the programs, you can transfer a DOS text file in which you have "listed" a BASIC program (see page 76 of the DOS manual) to a Pascal text file, edit the program using the Pascal editor, transfer it back, and EXEC it. Of course you could also originate the program text in the Pascal editor. Either way you can take advantage of the considerable power of the Pascal editor to build BASIC programs.

Going the other way, Pascal programmers can pick up high resolution graphics images created under DOS and transfer them to the Pascal operating system. The same is true of other DOS files but there is likely to be some additional work to do at the Pascal end, for example formatting random access text files into a record structure to be used by a Pascal program. Given the need to observe word boundaries in Pascal, this could mean a custom program for each DOS file translated unless you write a very flexible second stage formatting program. The need to observe word boundaries and disassemble DOS's free use of single bytes is (unfortunately, he says) well illustrated in 'puffin'.

Puffin: What Does it Do?  
=====

Puffin is a program to transfer arbitrary DOS files to Pascal files. Provision has been made for specifying the type of Pascal file to be created independent of the type of DOS file selected. For example, it is possible to force a tokenized BASIC file into a Pascal file even though the results

will be virtually meaningless. More useful examples were mentioned above. The Pascal file types which can be created are text, foto and data.

Puffin presents a menu of four commands to the user: C)atalog, D)isplay, T)ransfer and Q)uit.

Catalog assembles and displays a DOS directory. Given the volume number of a disk drive as specified in Pascal (page 171 in the Language Manual, page 276 in the Operating System Manual) the procedure searches the designated disk for the sectors of a DOS directory, assembles the information, and displays the results on the console screen. The displayed information includes the filename, filetype, sector length, location of the first track sector list and protection status of each file in the directory.

Display simply lists the contents of the last directory obtained by catalog. This is referred to as the current or working directory.

Transfer is the workhorse. It fetches the name of the DOS file to be transferred, the Pascal destination and the type of destination file. The transfer is initiated only if the DOS file is in the current directory and a valid Pascal file name is specified.

IO error protection is used extensively to trap egregious keyboard blunders and disk IO errors. The program will not run away from you and control is always returned to the main menu in a disciplined way.

#### Globals and the Main Man =====

The global declarations for PUFFIN define the Pascal representation of a DOS directory and set the stage for the three working commands in the program.

Note that all strings are defined in terms of previously defined constants which determine their length, for example:

```
didleng=30;  
did=STRING[didleng];
```

Also, note that all arrays are defined using constants which specify their bounds, types which specify the base type and sub-range types which are used to specify the range of an index into the array. For example:

```
TYPE  
  blocksize=512;  
  blockrange=0..blocksize;  
  blockbuffer=  
    PACKED ARRAY[1..blocksize] OF byte;  
VAR  
  block:blockbuffer;  
  blockindex:blockrange;
```

work together to specify what a 'blockbuffer' is and a pointer, 'blockindex', into the variable 'block'.

The use of constants increases readability and facilitates program modification. The use of sub-range types increases the security of the program in the sense that violating the sub-range bounds will cause either a compilation or execution error- the language will not permit you to commit a range error. In more complicated programs this is a very definite plus but even here the discipline is appropriate.

The declaration for a DOS directory entry, 'dosdirent', should be familiar to DOS enthusiasts. Note that there are essentially two variations on a directory entry "tagged" by the field 'dfkind'. One variant, when dfkind = volinfo, occupies only the zeroeth entry of the directory, and contains the unit number from which the directory was taken and the number of entries in the directory. The other variant is used to describe an actual entry, and includes, of course, the field 'dfkind' denoting the type of DOS file.

The main program opens with an initialization of the directory. The zeroeth entry is set up for the dfkind = volinfo variant, the number of entries is set to zero and the associated unit number for the directory is also set to zero. After this the program enters a loop which prints the command menus, fetches the desired command and executes it. The loop, and the program, terminates only upon selecting 'quit'.

#### Catalog =====

The structure of a DOS catalog is that of a linear linked list in which each node occupies one sector on diskette and consists of two fields, one a pointer to the location of the next sector, and the other a list of up

to seven directory entries. A Pascal declaration which describes this is:

```
dosnode=RECORD
  nextnode:link;
  entries:ARRAY[1..7] OF
    dosdirentry;
END;
```

where 'link' and 'dosdirentry' are as actually declared in PUFFIN.

If only the layout of a DOS directory followed this declaration byte for byte it would be a trivial matter to traverse the list, reading by turns each DOS directory sector and displaying them on the screen. Unfortunately the layout does not so conform and there arises the additional complication of having to reformat the information contained in a directory sector. Although not difficult, it is messy and involves moving bytes around. Moreover the main idea of traversing a linked list is still at the core of the algorithm.

How then do we traverse the list and read the directory? We need only know where the list begins, when the last sector has been reached, and whether the entries are active or deleted. Following the information in the DOS manual, we assume the directory begins on track 17, sector 15, and that the last sector has been reached when the link to the next sector points to track 0, sector 0. Furthermore, we assume that a directory entry is inactive when the pointer to the track number for the entry's track sector list (relative byte 0 of an entry) is either 0 or 255 (\$FF).

'Catalog' begins by obtaining the number of the drive containing the DOS diskette to be cataloged. If drive 0 is selected, control is returned to the main program. Otherwise initialization of the variables 'dirlink' and 'entrycount' is performed. Note that 'dirlink' now points to the first sector of the DOS directory.

The procedure now enters two nested WHILE loops which perform traversal of the DOS directory and transfer of the information to the variable 'dosdir'.

The outer loop is controlled by the BOOLEAN function 'eodir' which tests whether there is another directory sector to be read. The function returns FALSE if and only if the variable 'dirlink' points to track 0,

sector 0.

Within the loop the first task to be performed is to read into the variable 'dirsector' the disk sector pointed to by 'dirlink'. Assuming this is accomplished without error, the procedure initializes the variable 'sectorindex' and enters the inner loop of the traversal.

The inner loop is controlled by the BOOLEAN function 'eodirsector'. This function searches the current directory sector and returns FALSE if and only if it finds an active directory entry. The search begins at entry number 'sectorindex+1' and continues until an active entry is found or the end of the sector is reached, whichever comes first. The end of sector is reached when all 7 (=maxindex) potential entries have been inspected. Note that initialization of the variable 'sectorindex' insures that inspection starts with entry 1.

In addition to updating 'sectorindex', 'eodirsector' also updates 'entrybase'. The latter points into the variable 'dirsector' marking the first byte of information for a directory entry.

At this point in the inner loop, 'entrybase' points to the start of the next directory entry to be converted to Pascal format. This happens in three steps. First, the information to be converted (35=entrylength bytes long) is moved to the variable 'nextentry'; second, the variable 'entrycount' is updated; and third, the actual formatting is done by 'filldirentry'. Careful programmers will note that the variable 'nextentry' and the call to 'moveleft' which fills it can be eliminated if appropriate changes are made to filldirentry.

After exhausting the current directory sector, the procedure updates 'dirlink' and the test controlling the outer loop is performed. Finally, upon exiting the outer loop the zeroth entry of the directory is updated and a call to 'displaydir' is made to display the contents of the directory.

The procedure 'filldirentry' does the dirty work of formatting a DOS entry for Pascal. One important task it performs is to normalize the DOS file name to true ASCII by setting low the high bit of each character. For program security it also sets any

non-printing characters to blanks. After that it finds the leftmost trailing blank in the name field and sets the length of the name accordingly.

#### Displaydir =====

The procedure 'displaydir' is straight forward. A header is displayed, then entries are displayed one at a time. Provision is made for suspending the the display when the screen is filled and even for escaping to the main program. A variable 'cumsectors' keeps a running total of the number of occupied sectors.

The procedures 'displayheader' and 'displayentry' are declared global to 'displaydir' because they are called by 'transfer'.

The directory display takes advantage of Pascal's 80 column format. Those who wish to avoid having to use CNTRL-A for horizontal scrolling may wish to adjust the display procedures accordingly.

#### Transfer =====

The key to a DOS file is its track sector list and like the DOS catalog the list is a linear linked list. Nodes consist of two fields, a pointer to the next node (the continuation of the track sector list), and a list of pointers to the disk sectors allocated to the file up to a maximum of 122 (=maxlink) in a node. The declarations

```
CONST
  maxlink=122;
TYPE
  byterange=0..255;
  link=PACKED RECORD
    tracknum:byterange;
    sectnum:byterange;
  END;
  tslist=RECORD
    continuation:
      link;
    list:
      PACKED ARRAY[1..maxlink];
  END;
VAR
  currentlist:tslist;
```

correspond to this structure.

The structure of the track-sector list suggests the need for a traversal algorithm similar to that used in

'catalog', and appropriate tests for when the end of the list has been reached and so on. These are explained below.

'Transfer' starts off with two loops to find out what to transfer and where to send it. It elicits the name of the DOS file to transfer. If none is given, control is returned to the main program; otherwise, a loop is entered which is controlled by the BOOLEAN function 'searchdir'. This function returns TRUE if and only if the designated DOS file is in the working directory; if it returns TRUE it also returns the index number of the entry.

After this the procedure conveniently displays the directory information for the selected file and initializes the variable 'nextnode' to point to the first node (sector) of the file's track-sector list.

At this point the procedure calls 'getfiletype' to elicit the type of Pascal file the DOS file is to be transferred to. Three choices are available: text, foto and data.

The procedure 'filetype' returns a string 'suffix' and a variable 'filetype' of type 'pasfilekinds'. 'suffix' is appended to the name of the destination file, and 'filetype' controls the initialization and formatting to be performed.

Next the procedure elicits the name of the destination file and enters a WHILE loop under control of 'openfile' which attempts to open the destination file. 'openfile' traps illegal file names and otherwise prevents the program from crashing while trying to open the file. Note that it is the user's responsibility to see to it that the destination file will be written on a diskette other than the DOS source. The program is not protected against blunders of this type and recovery cannot be guaranteed. Take it from one who knows!

The procedure now initializes its key variables. The transfer buffers 'primpage' and 'sparepage' are filled with null characters; their associated position pointers are set to zero; and the variable 'relblock' which keeps track of the number of blocks written, is set to zero. If the destination file is a 'fotofile', the BOOLEAN variable 'fotoflag' is set to TRUE. If the destination file is a text file two



blocks of nulls are written to the head of the file; these blocks are used by the editor and are best set to nulls in operations such as this.

The buffers 'primage' and 'sparepage' are each two blocks long to facilitate transferring to Pascal text files. Arranging other types of file transfers to fit within these declarations is easy to do.

At last we enter the nested WHILE loops which comprise the actual transfer of information.

The outer loop is controlled by the BOOLEAN function 'eolist' which tests whether there is a continuation to the track-sector list. The function returns TRUE if and only if 'nextlink' points to track zero, sector zero. Note that initialization of 'nextlist' insures that the transfer of information begins correctly.

Within this loop the first task is to retrieve the node in the list pointed to by 'nextlink'. This is performed by the function 'get\_node' which returns FALSE if and only if the retrieval results in an IO error; otherwise it returns TRUE and stores the sector it gets in the variable 'currentnode'.

Having obtained the current node, 'transfer' initializes the variable 'linkindex' and enters the inner WHILE loop controlled by the function 'eonode'. This variable-function pair is very similar to the pair 'sectorindex' and 'eodirsector' used by the catalog procedure. 'eonode' returns FALSE if and only if it finds an active file sector and if it does it also returns the location of the sector in 'nextlink'. Active sectors are indicated by track-sector pointers which do not point to track zero, sector zero.

Continuing through the inner loop we make a call to 'readtrksec' which reads the file sector pointed to by 'nextlink' and stores the data in 'nextsector'. If all goes well the information in 'nextsector' is stuffed into 'primage' according to the format indicated by 'filetype'. If after exiting 'stuff' 'primage' is full, the data contained therein are written to the destination file by 'writeblocks'. Control is then returned to the inner loop test. When the inner loop is exited 'nextlink' is updated and

control reverts to the outer loop test.

#### Writeblocks =====

The task of writing the transfer buffer 'primage' is isolated in 'writeblocks'. Upon entry the variable 'pageptr', which points to the last character placed in 'primage', should be divisible by 'blocksize', and the variable 'blockcount' should contain the number of blocks to be written. If the call to the intrinsic 'blockwrite' fails to return a value equal to the number of blocks scheduled to be written, 'transfer' is aborted by a call to 'abornxfer'; otherwise 'relblock' is updated and control is returned to 'transfer'.

#### Three Flavors of Stuffing =====

The details of converting from DOS to Pascal format are hidden in the procedure 'stuff'.

The simplest stuffing is that which converts a DOS file to a Pascal data file. In this situation the DOS file is transferred byte for byte, sector for sector to the Pascal destination.

The assumption when transferring to a foto file is that the DOS source file is a binary file so that the first four bytes of file data are address and length information (presumably the address of one of the high res pages and the length of a hires image) which are important only to DOS. Consequently, the sub-procedure 'stufffoto' is constructed to ignore the first four bytes of the first sector of file data but transfer all succeeding sectors intact. The signal to ignore the first four bytes is relayed in the variable 'fotoflag'. Note that 'stufffoto' does not require a graphics image as input, only that it ignores four bytes.

'stufffoto' works as follows except when dealing with the first sector of the source: It first moves whatever is in the spare page into 'primage'; since we only read a sector at a time this is guaranteed to be at most one full sector. It then moves as much of the current sector, 's', as it can into the primary page. If this is a complete sector we exit 'stufffoto'; if it is less, the balance of 's' is moved to 'sparepage' and we exit 'stufffoto' with a full primary page which is then

cleared with a call to 'writeblocks'.

Converting to a text file is somewhat more complicated. We must make sure that the destination file follows the syntax for a text file if we expect to edit the file with the Pascal editor. 'Stufftext' insures compatibility with the system editor.

An editor compatible, Pascal text file can be defined from the top down as follows: A text file comprises two 512 byte blocks of header information followed by an arbitrary number of 1024 byte pages of text. Each page comprises a sequence of lines, and a line is a sequence of ASCII characters of the form

```
<DLE indent> <char>...<char> <CR>
```

where DLE and CR are ASCII 16 and 13 respectively, indent is the number of leading blanks+32 in the line, and 'char' denotes any printable ASCII character. Lines are not permitted to cross page boundaries, and the last line on a page is followed by a sequence of nulls to fill out the 1024 bytes. Note that this setup imposes the extremely mild condition that a Pascal line (but not an English sentence) can be a maximum of only 1024 bytes long.

'stufftext' transforms a DOS file into a Pascal file a sector at time taking care to observe line to page boundaries and converting all input characters to true ASCII.

The procedure begins with the conversion to ASCII, followed by conversion of all non-printing characters to nulls, followed by elimination of all null characters.

At this point the input sector 's', comprising a total of 'lengindex' valid characters, is ready to be transferred to the buffer 'primpage'. If only we were willing to assume that no input line will ever contain more than 256 valid characters we could transfer from 's' to 'primpage' directly. Unwilling to do this we first transfer 's' to 'sparepage' which allows us to deal with input lines of up to 1024 characters, the Pascal maximum.

The procedure now enters a WHILE loop which scans 'sparepage' for successive lines and transfers them to 'primpage'. The variables 'leadindex' and 'lagindex' and the byte oriented

function 'scan' are used to accomplish this. 'lagindex' points to the end of the last line transferred and is set to zero at the outset. The function 'scan' is used to find the next carriage return starting at the location just past 'lagindex'. And 'leadindex' is used to mark the location found by 'scan'. Thus leadindex-lagindex is the number of characters in the line, including the carriage return.

The WHILE loop is terminated when all characters in 'sparepage' have been scanned or when it is known that the next line to be transferred will cross a page boundary. Upon exiting the loop the balance of the information in 'sparepage' is shifted to the left to provide a clean start when 'stufftext' is re-entered.

#### Bridging the Gap =====

Doing yeoman duty in support of the heavy players is a small band of relatively low-level procedures and functions. Chief among these is 'readtrksec' which is the bridge from DOS to Pascal and summarizes the different ways these two operating systems look at disk data.

DOS of course treats a 256 byte sector as the basic block of data and looks at the disk as a collection of sectors within tracks, 16 sectors per track, 35 tracks to the disk. Pascal on the other hand deals in 512 byte blocks and organizes the disk into 280 blocks. Reading a DOS sector into Pascal entails reading the block which contains the desired sector and retaining the appropriate 256 bytes.

Interesting to me was the discovery that DOS sectors do not logically map sequentially to Pascal blocks. For example, block 0 contains (track 0) sectors 0 and 14, block 1 contains sectors 12 and 13 and block 7 contains sectors 1 and 15.

The function 'writetrksec' maps from Pascal to DOS but is not used here.

#### The Ultimate Perversion =====

Got your BASIC program up and running? OK, pass it over to Pascal using PUFFIN, compile it with your handy-dandy BASIC-to-6502 compiler (written in Pascal, of course), and

pass it back to DOS using Dana's program, HUFFIN. Off you go.

What's that you say? No such compiler?

Pour me another cup of coffee....

BLAISE AWAY!!!!

Dr. Wo

(listing begins on p.36)

# WAP DISK #29 CORRECTION by Bob Schmidt

I purchased Washington Apple Pi Disk #29 particularly because I was interested in the Telephone Dialer program. The program requires that a relay be connected to the game I/O socket, the procedure for which is described in the program REM statements. The REMS do not, however, specify all of the electrical values for the relay coil and I grew wary of this procedure since the Apple Reference Manual states that a buffer (another integrated circuit) is required "to drive other than TTL (IC) inputs (outputs)."

I found that the IC (74LS259) that drives the AN3 output of my Apple has current specifications such that any commercial relay I could find could make it an expensive fuse or, at best, create an overload condition for the IC. Accordingly, I recommend that you avoid the procedure in the Telephone Dialer REM statements.

Instead, you should use a suitable 7400 series IC or transistor. This plus a bit of wire and hardware will satisfy the buffer requirement for the game I/O socket. This addition will isolate your Apple from the "mistakes" of the outside world, will give the necessary current capability and, in the event of disaster, will serve as an inexpensive "fuse" (\$.25-\$.50) for short circuits. Once the buffer is in place, the relay described in the REMS can be added.

This is not a project for those of you who are not electrically inclined. Find a friend who is and swap your latest software masterpiece for some hardware help.

I've incorporated these additions into my own version of an isolated game I/O panel into which I can connect control devices without constantly substituting connectors into the motherboard game socket. I now have that warm feeling that I won't "zap" my Apple.

## MINI FLOPPY DISKS FOR APPLE DISK DRIVES

KYBE sgl side, dbl density, soft sector	28.50
KYBE dual side, dbl density, soft sector (flippy)	45.00
MAXELL sgl side, dbl density, soft sector	44.00

## PRINTWHEELS AND THIMBLES

QUME Std., AGT, and Diablo	6.40
QUME Special	7.90
NEC Spinwriter thimbles	13.95
88 char. metal printwheels	41.50
96 char. metal printwheels	49.95
Dual Plastic (Vydec type)	18.99

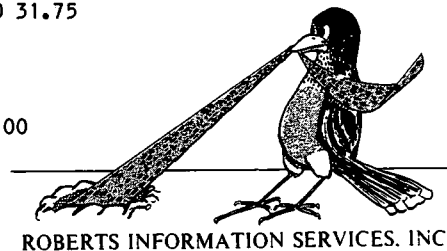
## PIN-FEED PAPER

9 1/2 x 11 15# 3500	27.25
9 1/2 x 11 20# 2700	25.95
12 x 8 1/2 15# 3200	31.36
12 x 8 1/2 18# 2600	31.36
14 7/8 x 11 15# 3000	32.19
14 7/8 x 11 20# 2400	31.75

## FORMS TRACTOR

Rutishauser, bidirectional 210.00  
**MODEM**

Lexlcon, LX-11 300 baud 145.00



**Roberts Information Services, Inc (ROBINS)**  
8306 Hilltop Road Fairfax, VA 22031 703/ 560-5900  
(across from the Post Office in Merrifield)  
Monday-Friday 9 to 5 Saturday 10 to 2  
We ship by Capitol Messengers and UPS

### 703/560-5900

# 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

dictionary with 2000 hard to spell words

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

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

current session  
interactive query updating

### includes text formatter

free with purchase of ms.speller

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



## Intelligent Computer Systems Corporation

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

# HIT PARADE by

John Alden

## RESULTS OF THE GAME BUYER'S SURVEY

Recently SIGAMES started a new service for all WAP members: A series of game buyer's guides.

This article is the first of the series. The buyer's guide (shown in the accompanying chart) is the result of a survey of SIGAMES members during the June meeting. In the following months, other series of games will be surveyed at the SIGAMES meetings.

The groupings will include adventure games, simulations, war games, arcade games, board games, sport games, space games, and puzzle games. Additionally, each grouping (except arcade & board games) can be subgrouped into text or graphical (low- or high-res) games.

Each category (or, subcategory) will be surveyed at the beginning of the SIGAMES meeting. The results will be presented at the next SIGAMES meeting and published the following month in the club newsletter.

The current plan is to publish each category separately. Thus, the board game and arcade game surveys will appear in separate issues of the newsletter.

Anyone who cannot attend the meetings and would like to contribute to the surveys should contact Al Gass (703) 371-3560 or John Alden (202) 686-1656.

New contributions will be incorporated into the appropriate survey and will be republished as soon as possible. (This is the only exception to publishing one survey at a time.)

The schedule for conducting the surveys is as follows:

MEETING	CATEGORY SURVEYED
June	Adventure (graphic)
July	War games (all)
August	Adventure (text)
September	Simulation (all)
October	Arcade
November	Space games (graphic)
December	Space games (text)
January	Sport games (all)
February	Puzzle games (all)

## CATEGORIES

The surveys are based on the following seven categories:

**ACTION:** The action is the series of events which form the plot or theme of the adventure game. It should move along rapidly, yet the adventurer should not have to defend his or her life when entering each new room or area. An adventure should not grind to a halt because the adventurer cannot locate the tool or object necessary to advance past an obstacle.

**GRAPHICS:** Graphics are essential to the success of the high-resolution adventure game. The graphics, as with the game itself, must be imaginative and well-done. Unfortunately, too often in the past, programmers have used either boring graphics (which detract from a superb game) or spectacular graphics to overcome a boring game.

**REALISM:** In a fantasy game????? Absolutely! In the August issue of "Creative Computing," Robert Plamondon stated that realism "means that none of the events breaks the character's 'willing suspension of disbelief.' Players can accept magic and dragons as part of the background of the fantasy world. They cannot accept worlds that turn upside down at night, outdoor human colonies on the sun, or personal clues displayed on billboards."

**RULES:** Traditionally, the worst aspect of most games has been the rules. If they were complete, they were written for cryptographers. If the rules were at all understandable, they were incomplete.

**BALANCE:** How quickly were you killed the last time you tried an adventure game? Balance refers to the capability of the program to act as a referee and as your opponent.

**STRATEGY:** What is the overall game plan? Is it to get the golden apple, rescue the princess, destroy the asteroid, or escape the island? Strategy is the planning and developing of game goals.

**TACTICS:** Hand-to-hand combat!!! you versus the computer dragon!!! Tactics is the step-by-step


process for a successful strategy.

**CUMM:** The cumulative score for each of the games surveyed.

**LIST \$:** The manufacturer's list price was inadvertently omitted from this survey. It will be a regular part of future surveys.

### RESULTS

Akalabeth by California Pacific Computer Co. was the best overall game. This game captured all categories but BALANCE and STRATEGY. The game winning the BALANCE category was the Prisoner by EduWare, while Automated Simulations' Hellfire Warrior captured the STRATEGY category.

The scoring for the surveys is based upon a "1" to "10" scale. A "10" is outstanding. A "5" is an average game and a "1" is abominable. 

### \* WASHINGTON APPLE PI \*\*\* GAME BUYER'S GUIDE \*

HI-RES ADVENTURES:	ACTION GRAPHICS	REALISM	RULES	BALANCE	STRATEGY	TACTICS	CUMM:
AKALABETH	8.00	9.00	9.00	9.00	8.00	8.00	8.57
ODYSSEY	7.25	7.00	8.00	7.75	7.25	6.00	7.29
PRISONER	5.89	6.22	5.67	5.00	5.89	8.44	6.43
DEATHMAZE 5000	7.00	8.00	8.00	7.00	4.00	5.00	6.00
WIZARD & PRINCESS	4.00	8.22	5.33	5.22	6.00	6.33	5.75
HELLFIRE WARRIOR	5.00	5.00	5.00	5.00	10.00	5.00	5.71
MORLOC'S TOWER	6.00	4.00	6.50	8.00	5.50	5.00	5.57
DATESTONE OF RYN	5.00	5.75	4.75	7.50	4.50	6.00	5.54
WILDERNESS CAMPAIG	5.00	6.33	6.67	5.00	5.00	5.67	5.52
BENEATH APPLE MAND	5.17	3.17	5.83	6.17	6.50	5.83	5.43
DUNGEON CAMPAIGN	4.33	2.33	5.33	5.00	3.33	1.67	3.71
MYSTERY HOUSE	1.80	4.00	4.00	4.00	3.60	4.00	3.26
TEMPLE OF APSHAI	4.50	1.33	2.17	2.67	1.83	1.67	2.21
MISSION: ASTEROID	1.40	3.00	2.20	1.60	1.80	0.60	1.71



cont'd. from p.2

the differences between a circular and a linear adventure. A circular adventure starts and ends at the same point. The linear adventure starts at one point and ends elsewhere. Along the way, you acquire items which will enable you to reach the end. If you haven't got the cracker before you reach the parrot on the other side of the bridge, you cannot get off the island. (No, I won't tell which adventure this is!)

'Beneath Apple Manor' is possibly the most primitive graphic adventure game still on the market. When playing the game, you have a choice of color or black & white. The color choice is done in low-res graphics and the black & white version is in hi-res. Isn't that sensible???? In the hi-res version, you are represented as a "y". The treasure is a "\$" and the Troll is a "T". BAM has a definite ending. You must find the 'Golden Apple'. Beware of substitutes!!! There is some magic and a selection of monsters. The ending is variable as are the rooms and treasure locations providing endless potential games. You may also select levels of difficulty on a 1 to 10 scale (10 being the roughest.) On the negative side, you cannot save the game. This is primarily a tactical game of the "Monty Haul" variety. If you refer to this month's game buyer's guide, SIGAME members surveyed consider BAM to be 5.43 with a 5 being average.

'The Wizard and the Princess' is a graphic example of a linear adventure. (for those of you who do not know me, the pun is unintentional???) You must find the way to the castle to rescue the princess from the wizard who holds her captive. The graphics are well done and the colors are vibrant. The puzzles are not easy but they are solvable with some effort. SIGAMES members voted


this game a 5.75 even though it is number 6 on Softalk magazine's Top Thirty list of best-sellers.

'Odyssey' was number 2 on the SIGAMES survey of graphic adventure games. It combines the best features of 'Beneath Apple Manor' and 'The Wizard and The Princess'. It is a linear adventure with a finite end but it is variable and can be saved. Done in hi-res, the colors are outstanding. You have puzzles to solve and treasures to find before you can purchase an army and a ship to leave the island. But, you must find the treasures quickly, because your army lives on its stomach and you must buy the food. Granted that you can bargain with the merchant. But be cautious, if you offer too low a price, the merchant will no longer sell the food. You must leave and hopefully return later. There are many involved strategies and tactics. Persons of all ages will find this game challenging. A highly recommended purchase.

'Akalabeth' stole top honors in the survey with an overall score of 8.57. It is a parallel development to 'Odyssey'. While the graphics representing the towns, the castles and the terrain are primitive, the dungeons are done in spectacular hi-res graphics. The game is variable but it cannot be saved. This is the precursor to Ultima.

'Ultima' was released during the early part of July. It is already number 19 on Softalk Magazine's Top Thirty bestsellers' list. The manufacturer bills this game as the new generation of Fantasy Role-Playing games (FRP's). The game spans not only a geographic area but space and time. You begin in the middle ages and time advances forward to the space era. The game is done in real time. It can be saved (but not in the

dungeons). There is a finite end. Along the way you will go on quests, slay monsters, rescue a princess, become a space ace, cast spells, and slay an evil wizard. You can ride on a horse, a raft, or cart while on an island. To travel over the sea you will find the frigate or the air-car necessary. The space shuttle will get you to the space station. Beware, monsters and villians abound. Over 25 monsters are in the dungeons and almost that many are on the surface. Weapons exist to destroy these monsters, but they are not always those which you would expect. The program is

so large that it is on a two-sided disk. One side is copy-protected and the other is not. The action is faster and smoother if you copy the unprotected side to a blank disk to store your character. The game defaults to one disk but you can toggle it to two disks from the main menu. One of the strongest recommendations I can make is to frequently save your game. This is done by entering "Q". The game is saved and action will continue from where you left off. One of the best games to be developed. A highly recommended purchase. 



**9174 Broken Oak Place  
Burke, VA 22015  
(703) 455-3432**

## **Disk Connection**

**WE HANDLE:**

Wabash..... \$26.00  
Maxell..... \$32.00  
Verbatim Data Life..... \$26.00

**ALSO AVAILABLE:**

Flip-Files..... \$22.00  
Plastic Pages  
(25 per Pkg)..... \$12.50  
Disk Saver Kits..... \$10.00

**WE CAN  
DELIVER TO  
WAP OR NOVAPPLE  
MEETINGS  
OR  
WE CAN SHIP UPS**

VIRGINIA RESIDENTS ADD 4% TAX

```

(*$$*)   cont'd. from p.31
(*$V-$)
PROGRAM puffin;
CONST
  maxunit  =12; (* maximum number for a pascal unit *)
  maxdir   =105; (* maximum number of entries in a DOS diskette directory *)
  maxlink  =122; (* maximum number of entries in a track sector list *)
  didleng  =30; (* maximum length of a DOS file name *)
  pidleng  =23; (* maximum length of a Pascal file name *)
  sidleng  = 5; (* maximum length of a Pascal file name suffix, e.g. ".TEXT" *)
  sectsize =256; (* size of a DOS sector *)
  blocksize=512;
  pagesize =1024; (* size of a pascal text page *)
  maxbyte  =255;

  dirtrack  =17; (* track number where a DOS directory resides *)
  firstdirsect=15; (* first sector of a DOS directory *)

TYPE
  byterange =0..maxbyte;
  sectrange =0..sectsize;
  dirrange  =0..maxdir;
  linkrange =0..maxlink;
  unitrange =0..maxunit;
  blockrange=0..blocksize;
  pagerange =0..pagesize;

  sectbuffer =PACKED ARRAY[byterange] OF byterange;
  blockbuffer=PACKED ARRAY[1..blocksize] OF byterange;
  pagebuffer =PACKED ARRAY[1..pagesize] OF byterange;

% link=PACKED RECORD (* used to designate track/sector combinations *)
  tracknum:byterange;
  sectnum:byterange;
  END;
  tslist=(* track sector list *)
  RECORD
    continuation:link;
    list:PACKED ARRAY[1..maxlink] OF link;
  END;

  did=STRING[didleng];
  pid=STRING[pidleng];
  sid=STRING[sidleng];

  dosfilekinds= (* DOS file types *)
    (volinfo,unknown,dfntext,dfinteger,applesoft,binary);
  pasfilekinds= (* some of the Pascal file types *)
    (textfile,fotofile,untyped);

  (* Pascal format for the information contained in a DOS directory entry *)
  dosdirentry=PACKED RECORD CASE dfkind:dosfilekinds OF
    volinfo: (* this is volume info *)
      (dunitnum:unitrange;
       dnumentries:dirrange);
    unknown,
    dfntext,
    dfinteger,
    applesoft,
    binary:
      (file_tsl:link; (* location of file's track-sector list*)
       locked:BOOLEAN; (* designates whether file is locked *)
       name:did;
       sectorcount:byterange); (* number of diskette sectors allocated *)
  END;

```

```

  dosdirectory=ARRAY[dirrange] OF dosdirentry;
  VAR
  dosdir:dosdirectory; (* current working DOS directory *)
  unitnum:unitrange;
  ioerror:INTEGER;
  ch:CHAR;

  FUNCTION readtrksec(unitnum:unitrange;
                     trksec:link;VAR sb:sectbuffer;VAR ioerror:INTEGER):BOOLEAN;
  (* reads sector number 'trksec.sectnum' from tracknumber 'trksec.tracknum'
   on disk drive number 'unitnum' *)
  VAR
  block:blockbuffer;
  blocknum,offset:INTEGER;
  BEGIN
  WITH trksec DO
  BEGIN
    (* compute half-block corresponding to desired sector *)
    IF (sectnum IN [0,15]) THEN blocknum:=sectnum
      ELSE blocknum:=15-sectnum;
    IF (odd(blocknum)) THEN offset:=256
      ELSE offset:=0;
    (* now compute blocknum off set from track 0 *)
    blocknum:=(blocknum DIV 2)+8*tracknum;
  END; (* WITH trksec DO *)
  (*$I-$)
  unitread(unitnum,block,sizeof(block),blocknum);
  (*$I+*)
  ioerror:=ioresult;
  IF NOT (ioerror=0) THEN readtrksec:=FALSE
  ELSE BEGIN
    (* write into sector buffer *)
    moveleft(block[offset+1],sb,sizeof(sectbuffer));
    readtrksec:=TRUE;
  END; (* IF...THEN...ELSE *)
  END;

  FUNCTION writetrksec(unitnum:unitrange;
                     trksec:link;VAR sb:sectbuffer;VAR ioerror:INTEGER):BOOLEAN;
  VAR
  blocknum,offset:INTEGER;
  block:blockbuffer;
  BEGIN
  (* see comments for 'readtrksec' *)
  WITH trksec DO
  BEGIN
    (* compute half-block corresponding to desired sector *)
    IF (sectnum IN [0,15]) THEN blocknum:=sectnum
      ELSE blocknum:=15-sectnum;
    IF (odd(blocknum)) THEN offset:=256
      ELSE offset:=0;
    (* now compute blocknum off set from track 0 *)
    blocknum:=(blocknum DIV 2)+8*tracknum;
  END; (* WITH trksec DO *)
  (*$I-$)
  unitread(unitnum,block,sizeof(block),blocknum);
  (*$I+*)
  ioerror:=ioresult;
  IF NOT (ioerror=0) THEN writetrksec:=FALSE
  ELSE BEGIN
    moveleft(sb,block[offset+1],sizeof(sectbuffer));
    (*$I-$)
    unitwrite(unitnum,block,sizeof(block));
    (*$I+*)
    ioerror:=ioresult;
  END;

```

```

        writetrksec:=ioerror=0;
    END;
END;

FUNCTION searchdir(target:did;VAR index:dirrange):BOOLEAN;
VAR
    found:BOOLEAN;
BEGIN
    found:=FALSE;
    index:=dosdir[0].dumentries;
    WHILE NOT (found OR (index=0)) DO
        BEGIN
            found:=target=dosdir[index].name;
            index:=index-1;
        END;
    IF found THEN index:=index+1;
    searchdir:=found;
END;

FUNCTION stoi:INTEGER;
VAR
    ch:CHAR;
    x:INTEGER;
BEGIN
    x:=0;
    read(ch);
    WHILE ch IN ['0'..'9'] DO
        BEGIN
            x:=10*x+(ord(ch)-ord('0'));
            read(ch);
        END;
    writeln;
    stoi:=x;
END;

FUNCTION get_unit_num(VAR unitnum:unitrange):BOOLEAN;
VAR
    un:INTEGER;
BEGIN
    REPEAT
        writeln;
        writeln('Enter the unitnum number [4,5,9..12] of the disk drive containing');
        writeln('the DOS diskette to be cataloged. Enter 0 to escape. ');
        writeln;
        write('>> ');
        un:=stoi;
        IF NOT (un IN [0,4,5,9..12]) THEN writeln(chr(7));
    UNTIL un IN [0,4,5,9..12];
    unitnum:=un;
    get_unit_num:=(un<>0);
END;

PROCEDURE capitalize(VAR line:STRING);
CONST
    ordsmla=97;
    ordsmlz=122;
    shiftcase=32;
VAR
    index:0..maxbyte;
BEGIN
    FOR index:=1 TO length(line) DO
        IF line[index] IN [chr(ordsmla)..chr(ordsmlz)]
            THEN line[index]:=chr(ord(line[index])-shiftcase);
    END;

FUNCTION getpasid(VAR name:pid):BOOLEAN;

```

37

```

BEGIN
    writeln;
    writeln('Enter the name of the Pascal destination file,');
    writeln('or enter <RET> to exit:');
    writeln;
    write('>>');
    readln(name);
    IF (length(name)=0) THEN getpasid:=FALSE
    ELSE BEGIN
        capitalize(name);
        getpasid:=TRUE;
    END;
END;

FUNCTION getdosid(VAR name:did):BOOLEAN;
BEGIN
    writeln;
    writeln('Enter the name of the DOS file to transfer,');
    writeln('or enter <RET> to exit:');
    writeln;
    write('>>');
    readln(name);
    IF (length(name)=0) THEN getdosid:=FALSE
    ELSE BEGIN
        capitalize(name);
        getdosid:=TRUE;
    END;
END;

PROCEDURE getfiletype(VAR suffix:sid;VAR filetype:pasfilekinds);
BEGIN
    writeln;
    writeln('Transfer to a:');
    writeln;
    writeln('T)ext file, F)oto file, or D)ata (binary) file?');
    writeln;
    write('>> ');
    read(keyboard,ch);
    WHILE NOT (ch IN ['t','f','d','T','F','D']) DO
        BEGIN write(chr(7));read(keyboard,ch); END;
    writeln(ch);
    CASE ch OF
        'T','t':BEGIN suffix='.TEXT';filetype:=textfile; END;
        'F','f':BEGIN suffix='.FOTO';filetype:=fotofile; END;
        'D','d':BEGIN suffix='';filetype:=untyped; END;
    END;
END;

PROCEDURE printmenu;
CONST
    cleoln=29;
BEGIN
    gotoxy(0,0);
    write(chr(cleoln),'C)atalog, D)isplay, T)ransfer, Q)uit?');
END;

PROCEDURE readcommand(VAR ch:CHAR);
BEGIN
    read(keyboard,ch);
    WHILE NOT(ch IN ['C','c','D','d','T','t','Q','q']) DO
        BEGIN
            write(chr(7));
            read(keyboard,ch);
        END;
    writeln;
END;

```

```

PROCEDURE displayentry(de:dosdirent);
BEGIN
  WITH de DO
  BEGIN
    write(name,'',(didleng-length(name)+1));
    CASE dfkind OF
      dftext:write('text':6);
      dfinteger:write('int':6);
      applesoft:write('soft':6);
      binary:write('bnry':6);
      unknown:write('unkn':6);
    END;
    IF locked THEN write('yes':8)
    ELSE write('no':8);
    write(sectorcoun:9);
    writeln(filetsl.tracknum:6,'-',filetsl.sectnum:3);
  END;
END;

PROCEDURE displayheader;
BEGIN
  write('File Name');
  write('Type':((didleng-length('file name'))+7));
  write('Locked':8);
  write('Sectors':9);
  writeln('TSL link':10);
END;

PROCEDURE displaydir;
CONST
  cleos=11;
  esc=27;
  maxlines=21;
VAR
  cumsectors:INTEGER;
  count:dirrange;
BEGIN
  page(output);
  gotoxy(0,1);
  cumsectors:=0;
  IF dosdir[0].dnumentries=0 THEN writeln('The working directory is empty!')
  ELSE BEGIN
    displayheader;
    FOR count:=1 TO dosdir[0].dnumentries DO
    BEGIN
      displayentry(dosdir[count]);
      cumsectors:=cumsectors+dosdir[count].sectorcount;
      IF (count MOD maxlines)=0 THEN
      BEGIN
        write('Type <RET> to continue, <ESC> to stop ');
        read(keyboard,ch);
        IF ch=chr(esc) THEN exit(displaydir)
        ELSE BEGIN gotoxy(0,2);write(chr(cleos)); END;
      END;
    END;
    write(dosdir[0].dnumentries,' files on disk, ',cumsectors,' sectors in use');
  END;
END;

PROCEDURE catalog;
CONST
  nextlink = 1; (* relative byte 1 of directory sector is link to
                 next directory sector *)

```

```

zerobase =11; (* first byte of file info in a directory sector *)
entrylength=35; (* DOS directory entries occupy 35 bytes *)
mark =maxbyte; (* directory entries which have been deleted are 'marked'
                in (relative) byte zero *)
maxindex = 7; (* maximum of 7 directory entries in a sector *)

space= 32; (* ASCII space *)
tilde=126; (* ASCII tilde *)
TYPE
  indexrange=0..maxindex;
  entrybuffer=PACKED ARRAY[1..entrylength] OF byterange;
VAR
  sectorindex:indexrange;
  entrybase:byterange;
  dir_link:link;
  dir_sector:sectbuffer;
  nextentry:entrybuffer;
  entrycount:dirrange;
FUNCTION eodir(dirlink:link):BOOLEAN;
BEGIN
  WITH dirlink DO
    eodir:=(sectnum=0) AND (tracknum=0);
  END;
PROCEDURE fill_dir_entry(VAR de:dosdirent;VAR eb:entrybuffer);
CONST
  linkoffset = 1; (* relative byte zero for an entry gives the location of its
                  track-sector list *)
  kindoffset = 3; (* relative byte 2 designates the file type of the entry *)
  nameoffset = 4; (* relative byte 3 is the beginning of the file name *)
  countoffset=34; (* relative byte 33 is the sector count (MOD sectsize) for
                  the file *)
  lockbit =128; (* locked files have the high bit of the file type byte set *)
VAR
  j,kind:byterange;
  nonblank:0..didleng;
BEGIN
  WITH de DO
  BEGIN
    filetsl.tracknum:=eb[linkoffset];
    filetsl.sectnum:=eb[linkoffset+1];
    kind:=eb[kindoffset];
    IF NOT ((kind MOD lockbit) IN [0,1,2,4]) THEN dfkind:=unknown
    ELSE CASE (kind MOD lockbit) OF
      0:dfkind:=dftext;
      1:dfkind:=dfinteger;
      2:dfkind:=applesoft;
      4:dfkind:=binary;
    END;
    IF ((kind DIV lockbit)=1) THEN locked:=TRUE
    ELSE locked:=FALSE;
    FOR j:=0 TO (didleng-1) DO
    BEGIN
      (* set the high bit low to get true ASCII *)
      eb[nameoffset+j]:=eb[nameoffset+j] MOD 128;
      (* eliminate any weird characters *)
      IF NOT (eb[nameoffset+j] IN [space..tilde]) THEN eb[nameoffset+j]:=space;
    END;
    (* find the leftmost trailing blank in the name field *)
    nonblank:=scan(-didleng,<>' ',eb[nameoffset+didleng-1]);
    (* non_blank=0 if and only if no trailing blanks *)
    (* initialize the length of 'name' *)
    (*$R-*)
    name[0]:=chr(didleng-nonblank);

```



```

(*$R+*)
(* finally move in the name *)
moveleft(eb[nameoffset],name[1],length(name));
sectorcount:=eb[countoffset];
END; (* WITH de DO *)
END; (* filldirectory *)

FUNCTION eodirsector(VAR index:indexrange;
                    VAR dirsector:sectbuffer;VAR entrybase:byterange):BOOLEAN;
VAR
  nofile:BOOLEAN;
BEGIN
  nofile:=TRUE;
  WHILE (nofile AND (index<maxindex)) DO
    BEGIN
      index:=index+1;
      entrybase:=zerobase+(index-1)*entrylength;
      nofile:=(dirsector[entrybase] IN [0,mark]);
    END;
    eodirsector:=nofile;
  END;

BEGIN (* catalog *)
  page(output);
  IF NOT getunitnum(unitnum) THEN exit(catalog);
  WITH dir_link DO
    BEGIN
      tracknum:=dirtrack;
      sectnum:=firstdirsect;
    END;
    entrycount:=0;
    WHILE NOT eodir(dir_link) DO
      BEGIN
        IF NOT readtrksec(unitnum,dir_link,dir_sector,ioerror)
        THEN BEGIN writeln('ioerror ',ioerror,' reading directory');
                exit(catalog);
              END
            ELSE BEGIN
                sectorindex:=0;
                WHILE NOT eodirsector(sectorindex,dir_sector,entrybase) DO
                  BEGIN
                    moveleft(dir_sector[entrybase],nextentry,entrylength);
                    entrycount:=entrycount+1;
                    filldirectory(dosdir[entrycount],nextentry);
                  END;
                END; (*IF...THEN...ELSE *)
                WITH dir_link DO
                  BEGIN
                    tracknum:=dir_sector[nextlink];
                    sectnum:=dir_sector[nextlink+1];
                  END;
                END;
                WITH dosdir[0] DO
                  BEGIN
                    dnumentries:=entrycount;
                    dunitnum:=unitnum;
                  END;
                displaydir;
              END; (* catalog *)

PROCEDURE transfer;
TYPE
  ffile=FILE;
VAR
  dosname :did;
  pasname :pid;

```

```

suffix      :sid;
dirindex    :dirrange;
linkindex   :linkrange;
nextlink,
nextnode    :link;
nextsector  :sectbuffer;
currentnode :tslist;
ioerror     :INTEGER;
pasfile     :ffile;
primpage,
sparepage   :pagebuffer;
pagepnt,
sparepnt    :pagerange;
relblock    :INTEGER;
filetype    :pasfilekinds;
photoflag   :BOOLEAN; (* flag for shifts of size 'binaryoffset' *)

```

```

PROCEDURE abortxfer(ioerror:INTEGER);
BEGIN
  writeln;
  writeln('IO ERROR ',ioerror);
  writeln('EXITING TRANSFER');
  (*$I-*)
  close(pasfile,purge);
  (*$I+*)
  exit(transfer);
END;

```

```

FUNCTION openfile(name:pid;VAR f:ffile;VAR ioerror:INTEGER):BOOLEAN;
BEGIN
  (*$I-*)
  rewrite(f,name);
  ioerror:=ioresult;
  (*$I+*)
  openfile:=ioerror=0;
END;

```

```

FUNCTION eolist(next:link):BOOLEAN;
BEGIN
  WITH next DO
    eolist:=((tracknum=0) AND (sectnum=0));
  END;

```

```

FUNCTION get_node(location:link;VAR listdata:tslist):BOOLEAN;
CONST
  contoffset= 1; (* beginning of continuation link *)
  contleng = 2; (* length of continuation info *)
  listoffset= 12; (* beginning of list of track sector links *)
  listleng =244; (* length of list data *)

```

```

VAR
  sb:sectbuffer;
  i:linkrange;
BEGIN
  IF NOT (readtrksec(dosdir[0].dunitnum,location,sb,ioerror)) THEN get_node:=FALSE
  ELSE WITH listdata DO
    BEGIN
      continuation.tracknum:=sb[contoffset];
      continuation.sectnum:=sb[contoffset+1];
      FOR i:=1 TO maxlink DO
        BEGIN
          list[i].tracknum:=sb[listoffset+(i-1)*contleng];
          list[i].sectnum:=sb[listoffset+(i-1)*contleng+1];
        END;
      get_node:=TRUE;
    END;
  END;

```

```

FUNCTION eonode(VAR linkindex:linkrange;VAR tslink:link):BOOLEAN;
VAR
  emptysector:BOOLEAN;
BEGIN
  emptysector:=TRUE;
  WHILE ((linkindex<maxlink) AND emptysector) DO
    BEGIN
      linkindex:=linkindex+1;
      WITH currentnode.list[linkindex] DO
        emptysector:=(tracknum=0) AND (sectnum=0)
      END;
    IF NOT emptysector THEN tslink:=currentnode.list[linkindex];
    eonode:=emptysector;
  END;

PROCEDURE writeblocks(VAR pb:pagebuffer;pbpnr:pagerange;
  blockcount:INTEGER;VAR relblock:INTEGER);
BEGIN
  (***** note: pbpnr should be divisible by blocksize upon entry *)
  (*$I-* )
  pbpnr:=pbpnr-(blockcount*blocksize)+1;
  IF (blockwrite(pasfile,pb[pbpnr],blockcount,relblock)=blockcount)
    THEN relblock:=relblock+blockcount
  (*$I+* )
  ELSE abortxfer(ioerror);
END;

PROCEDURE stuff(VAR s:sectbuffer;VAR p:pagebuffer;
  VAR pagepnr:pagerange;filetype:pasfilekinds);
PROCEDURE stufftext;
CONST
  hbit=128;
  asciicr=13; (* ASCII carriage return *)
  space=32; (* ASCII space *)
  tilde=126; (* ASCII tilde *)
VAR
  lengindex,nextnull:sectrange;
  leadindex,lagindex:pagerange;
  cr,null:CHAR;
  primfull,endofspare:BOOLEAN;
BEGIN
  (* zero the high bits to get true ASCII *)
  FOR lengindex:=0 TO maxbyte DO
    BEGIN
      (* zero the high bit of s[lengindex] *)
      s[lengindex]:=s[lengindex] MOD hbit;
      (* eliminate weird characters by setting to null *)
      IF NOT (s[lengindex] IN [space..tilde,asciicr]) THEN s[lengindex]:=0;
    END;

  (* squeeze out the middle null characters *)
  null:=chr(0);
  lengindex:=sectsize;
  nextnull:=scan(lengindex,=null,s);
  WHILE (nextnull<lengindex) DO
    BEGIN
      moveleft(s[nextnull+1],s[nextnull],lengindex-nextnull-1);
      lengindex:=lengindex-1;
      nextnull:=scan(lengindex,=null,s);
    END;

  moveleft(s,sparepage[sparepnr+1],lengindex);
  sparepnr:=sparepnr+lengindex;
  endofspare:=FALSE;
  primfull:=FALSE;

```

```

  lagindex:=0;
  cr:=chr(asciicr);

  WHILE NOT (endofspare OR primfull) DO BEGIN
    leadindex:=scan((sparepnr-lagindex),=cr,sparepage[lagindex+1]);
    IF (leadindex=(sparepnr-lagindex)) THEN endofspare:=TRUE
    ELSE IF ((leadindex+pagepnr+1) > pagesize) THEN primfull:=TRUE
    ELSE BEGIN
      moveleft(sparepage[lagindex+1],primpage[pagepnr+1],leadindex+1);
      pagepnr:=pagepnr+leadindex+1;
      lagindex:=lagindex+leadindex+1;
      endofspare:=(lagindex=sparepnr);
    END;
  END;

  IF primfull THEN pagepnr:=pagesize;
  moveleft(sparepage[lagindex+1],sparepage,sparepnr-lagindex);
  sparepnr:=sparepnr-lagindex;

END;(* stufftext *)

PROCEDURE stufffoto;
CONST
  fotoffset=4; (* four bytes of DOS address junk in the first sector *)
BEGIN
  IF fotoflag (* first foto sector *)
    THEN BEGIN
      moveleft(s[fotoffset],primpage,sectsize-fotoffset);
      pagepnr:=sectsize-fotoffset;
      fotoflag:=FALSE;
    END
  ELSE BEGIN
    moveleft(sparepage,primpage[pagepnr+1],sparepnr);
    pagepnr:=sparepnr+pagepnr;
    sparepnr:=0;
    IF ((pagepnr+sectsize) <= pagesize) (* i.e., enough room for a sector *)
      THEN BEGIN
        moveleft(s,primpage[pagepnr+1],sectsize);
        pagepnr:=pagepnr+sectsize;
      END
    ELSE BEGIN
      (* move as much as possible into the primary page *)
      moveleft(s,primpage[pagepnr+1],pagesize-pagepnr);
      (* move the rest into the spare page *)
      (* begin by updating sparepnr *)
      sparepnr:=sectsize-(pagesize-pagepnr);
      moveleft(s[pagesize-pagepnr],sparepage,sparepnr);
      (* update pagepnr to end of page *)
      pagepnr:=pagesize;
    END;
  END;

END;

BEGIN
  IF (filetype=textfile) THEN stufftext
  ELSE IF (filetype=fotofile) THEN stufffoto
  ELSE BEGIN
    moveleft(s,p[pagepnr+1],sectsize);
    pagepnr:=pagepnr+sectsize;
  END;
END;

BEGIN
  page(output);
  IF NOT (getdosid(dosname)) THEN exit(transfer);
  WHILE NOT (searchdir(dosname,dirindex)) DO

```

```

BEGIN
  writeln;
  writeln(dosname,' not in current dosdir');
  IF NOT (getdosid(dosname)) THEN exit(transfer);
END;
writeln;
displayheader;
displayentry(dosdir[dirindex]);
nextnode:=dosdir[dirindex].file_tsl;

getfiletype(suffix,filetype);
IF NOT (getpasid(pasname)) THEN exit(transfer);
WHILE NOT (openfile(concat(pasname,suffix),pasfile,ioerror)) DO
  BEGIN
    writeln;
    writeln('IO error ',ioerror,' opening ',concat(pasname,suffix));
    IF NOT (getpasid(pasname)) THEN exit(transfer);
  END;

(* initialize the page buffers and associated pointers *)
fillchar(primpage,pagesize,chr(0));
fillchar(sparepage,pagesize,chr(0));
pagepnr:=0;
sparepnr:=0;
relblock:=0;

IF (filetype=fotofile) THEN fotoflag:=TRUE
ELSE IF (filetype=textfile) THEN
  (* write two header blocks of nulls *)
  BEGIN
    relblock:=blockwrite(pasfile,primpage,1,relblock)+relblock;
    relblock:=blockwrite(pasfile,primpage,1,relblock)+relblock;
  END;

41 WHILE NOT eolist(nextnode) DO
  IF NOT (get_node(nextnode,currentnode)) THEN abortxfer(ioerror)
  ELSE BEGIN
    linkindex:=0;
    WHILE NOT eonode(linkindex,nextlink) DO
      IF NOT (readtrksec(dosdir[0].dunitnum,nextlink,nextsector,ioerror))
      THEN abortxfer(ioerror)
      ELSE BEGIN
        stuff(nextsector,primpage,pagepnr,filetype);
        IF (pagepnr=pagesize) THEN
          BEGIN
            writeblocks(primpage,pagepnr,2,relblock);
            pagepnr:=0;
            fillchar(primpage,pagesize,chr(0));
          END;
        END;
        nextnode:=currentnode.continuation;
      END;

(* pick up anything in the spare page *)
moveleft(sparepage,primpage[pagepnr+1],sparepnr);
pagepnr:=pagepnr+sparepnr;
(* if page is partially full it needs to be written *)
IF (pagepnr>0) THEN (* note: pagepnr<pagesize will be true here *)
  BEGIN
    pagepnr:=blocksize*(1+(pagepnr DIV blocksize));
    writeblocks(primpage,pagepnr,(pagepnr DIV blocksize),relblock);
  END;

IF ((filetype=textfile) AND odd(relblock)) THEN
  BEGIN
    fillchar(primpage,pagesize,chr(0));

```

```

relblock:=blockwrite(pasfile,primpage,1,relblock)+relblock;
END;

writeln;
writeln(dosname,' transferred to ',concat(pasname,suffix));
writeln(relblock,' blocks transferred');
close(pasfile,lock);
END;

BEGIN
  WITH dosdir[0] DO
    BEGIN dfkind:=volinfo; dnumentries:=0; dunitnum:=0; END;
  page(output);
  gotoxy(0,5);
  writeln('Welcome to PUFFIN!');
  REPEAT
    printmenu;
    readcommand(ch);
    CASE ch OF
      'c','C':catalog;
      'd','D':displaydir;
      't','T':transfer;
    END;
  UNTIL ch IN ['Q','q'];
END.

```



# THE NATIONAL COMPUTER SHOWS

# HAVE WE GOT A PROGRAM FOR YOU IN '81

Attend the biggest public computer shows in the country. Each show has 100,000 square feet of display space featuring over 50 Million Dollars worth of software and hardware for business, industry, government, education, home and personal use.

You'll see computers costing \$150 to \$250,000 including mini and micro computers, software, graphics, data and word processing equipment, telecommunications, office machines, electronic typewriters, peripheral equipment, supplies and computer services.

All the major names are there including; IBM, Wang, DEC, Xerox, Burroughs, Data General, Qantel, Nixdorf, NEC, Radio Shack, Heathkit, Apple, RCA, Vector Graphic, and Commodore Pet. Plus, computerized video games, robots, computer art, electronic gadgetry, and computer music to entertain, enthrall and educate kids, spouses and people who don't know a program from a memory disk.

Don't miss the Coming Of The New Computers - Show Up For The Show that mixes business with pleasure. Admission is \$5 for adults and \$2 for children under 12 when accompanied by an adult.

### Ticket Information

Send \$5 per person with the name of the show you will attend to National Computer Shows, 824 Boylston Street, Chestnut Hill, Mass. 02167. Tel. 617 739 2000. Tickets can also be purchased at the show.

### THE SOUTHWEST COMPUTER SHOW

DALLAS

Dallas Market Hall  
2200 STEMMONS FRWY  
AT INDUSTRIAL BLVD

THURS-SUN  
APRIL 9-12  
10 AM TO 7 PM

### THE MID-WEST COMPUTER SHOW

CHICAGO

McCormick Place  
SCHOESSLING HALL  
23RD & THE LAKE

THURS-SUN  
SEPTEMBER 10-13  
10 AM TO 7 PM

### THE MID-ATLANTIC COMPUTER SHOW

WASHINGTON, DC  
DC Armory/Starplex

2001 E. CAPITOL ST. SE  
(E CAP ST EXIT OFF I295  
- KENILWORTH FRWY)  
ACROSS FROM RFK STADIUM

THURS-SUN  
SEPTEMBER 24-27  
10 AM TO 7 PM

### THE NORTHEAST COMPUTER SHOW

BOSTON

Hynes Auditorium  
PRUDENTIAL CENTER

THURS-SUN  
OCTOBER 15-18  
10 AM TO 7 PM

### THE SOUTHEAST COMPUTER SHOW

ATLANTA

Atlanta Civic Center  
395 PIEDMONT AVE NE AT  
RALPH MCGILL BLVD

THURS-SUN  
OCTOBER 29-NOVEMBER 1  
10 AM TO 7 PM

-----  
 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 33   | Accounting                 | ( ) |
| Volume 2  | Utilities II       | ( ) | Volume 34   | Solar Tutor                | ( ) |
| Volume 3  | Games I            | ( ) | Volume 35   | Garden Management          | ( ) |
| Volume 4  | Games II           | ( ) | Volume 36   | Games XII                  | ( ) |
| Volume 5  | Games III          | ( ) | Volume 37   | Utilities IX               | ( ) |
| Volume 6  | Games IV           | ( ) | Volume 38   | Games XIII                 | ( ) |
| Volume 7  | Games V            | ( ) | Volume 39   | IAC VII                    | ( ) |
| Volume 8  | Utilities III      | ( ) | Volume 40   | IAC VIII                   | ( ) |
| 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:     |                            |     |
| Volume 24 | Games IX           | ( ) | Volume 300  | ATTACH BIOS                | ( ) |
| Volume 25 | Utilities VII      | ( ) | Volume 301  | PIG1:                      | ( ) |
| Volume 26 | Stocks/Investments | ( ) | Volume 302  | PIG2:                      | ( ) |
| Volume 27 | Math               | ( ) | Volume 303  | PIG3:                      | ( ) |
| Volume 28 | Planetfinder       | ( ) | Volume 304  | PIG4:                      | ( ) |
| Volume 29 | Utilities VIII     | ( ) | Volume 305  | PIG5:                      | ( ) |
| Volume 30 | Games X            | ( ) | Volume 306  | PIG6:                      | ( ) |
| Volume 31 | Plot Utilities     | ( ) |             |                            |     |
| Volume 32 | Games XI           | ( ) | *Vol. 181   | required with these disks. |     |

-----  
 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