$1

Washington Apple Pi

Volume 1 October 1979 Number 9

Highlights

Multiprocessing With the APPLE page 6

by Bruce F. Field — Adding an external micro to the APPLE II

Plotpourri page 12

by Samuel S. Cottrell — Isometric 3D Plots show off APPLE's graphics

In This Issue

ComputerLand® and apple II

For the best in personal computing

SOFTAPE™ Personal Software™

D. C. Hayes Associates, Inc.
MICROCOMPUTER PRODUCTS

CENTRONICS®

Mountain Hardware, Inc.

Integral Data Systems, Inc.

MUSE

THE SOURCE
Information Utility

PDI

houston instrument

Heuristics INC.

Automated Simulations

SANYO

ComputerLand®
We Know Small Computers.
ComputerLand/Tysons Corner
8411 Old Courthouse Road at Rt. 123 — 893-0424

VISA

master charge

Officers & Staff

President	**- John L. Moon**	**(202) 332-9102**
Vice President	**- Bernard Urban**	**(301) 229-3458**
Treasurer	**- Robert Peck**	**(301) 770-1954**
Secretary	**- Genevie Urban**	**(301) 229-3458**
Members-at-Large	**- Mark Crosby**	**(202) 488-1979**
	- Sue Eickmeyer	**(301) 953-7355**
	- Sandy Greenfarb	**(301) 674-5982**
Editor	**- Bernard Urban**	**(above)**
Associate Editor	**- Mark Crosby**	**(above)**
Librarian	**- David Morganstein**	**(301) 972-4263**

Washington Apple Pi
PO Box 34511
Washington, D.C. 20034

PRESIDENT'S MESSAGE

[Ed. Note: John Moon was unable to present you with a message this month as he has been traveling. In place of the President's Message we are printing a note of general interest to all of you as APPLE users. mc]

Through a phone call to our Sandy Greenfarb from Val Golding of Call-A.P.P.L.E., Sandy learned that Jim Hoyt of Apple Computer, Inc., had commissioned Val to pull together individuals to attend a two-day meeting in San Francisco on October 27 and 28. Val was to choose from user groups on the basis of membership and geographic dispersion. Val invited us to choose a representative to attend at Apple's expense.

Sandy alerted John Moon and a special meeting of both Apple Pi and NOVAPPLE officers was called. After lengthy discussion, the officers decided to choose our representative by secret ballot. John Moon disqualified himself on the basis of conflict of interest. A vote was taken and Bernie Urban was elected. He asked for suggestions from the group on appropriate items to bring to the upcoming meeting and will be asking for input from our readers (see EDITORIAL).

Val said that invitations were given to user groups from Los Angeles, San Francisco, Seattle, Boston, Philadelphia, Detroit, Houston and Washington, DC. Although the exact purposes of the meeting are not clear, it is believed that the primary purpose of the meeting will be to organize and prepare an agenda for a meeting/convention to be held in March of 1980 at or simultaneously with the West Coast Computer Faire. At that meeting, speakers and attendees might discuss such items as: the merits of establishing, perhaps incorporating a National Apple Users' Group. Sandy indicated that another topic might center on the creation of a National Digest of User Group Newsletters. Still another might establish Special Interest Groups (such as Computer Assisted Instruction).

We hope to have much to report to you all after that meeting. Look for it in the next issue!

MINUTES

The meeting of September 29 was called to order at 9:40 by President John Moon. The following items of interest were covered:

1. Bulk purchasing of Memorex disks. 200 disks have been ordered and will be available for sale to members. More will be ordered. A purchase request sheet was passed around.

2. Hersch Pilloff reported on the group purchases of the Paper Tiger (IDS 440) printer. He reported a mail order price of: 1 - 9 units, 10% discount; 10 or more, 12% discount. Also, 2% additional discount if prepaid. The number of orders by Oct. 15 will determine the discount price for the following twelve months. At present he has orders for 5 units.

3. There was a discussion of the various characteristics of the ROMPLUS + board, and member experience with the board to date.

4. It was reported that Computerland (Gaithersburg) was offering a one-year subscription to Washington Apple Pi Newsletter with the purchase of an APPLE.

5. David Morganstein led a discussion of how our library should function. Some areas covered were the location of the library, software review facility, and whether there should be a charge for disk copying. A motion was made, amended and carried as amended to charge members $1.00 per disk side (or tape equivalent). A motion was also carried to create a Library Committee. David Morganstein and Sue Eickmeyer head that committee and volunters to assist were requested.

The meeting adjourned at 11:15 into informal groups and the hands-on APPLE session.

Minutes of NOVAPPLE for September 12, 1979

The entire meeting was devoted to a demonstration of the capabilities of "The Source". This program was presented by Steve Shank and Jim Clark. Approximately 50 persons attended. Since this program must be seen to appreciate its capabilities, you may contact "The Source", Telecomputing Corporation of America, 1616 Anderson Road, McLean, Virginia 22102 for literature on the program.

Minutes of the September 28 meeting:

The meeting was opened at 7:45 pm by the president. He announced that the next meeting of NOVAPPLE will be held at Computers Plus, Inc., at their new address: 6120 Franconia Road, Alexandria, VA 22310. Their new telephone number is (703) 971-1996. This will begin the new schedule of second Wednesdays at Computers Plus and fourth Thursdays at Computerland of Tysons Corner. We hope to introduce the owners and maybe demo some new software. Nominations, which were begun at Computerland, will be open again at Computers Plus, so we may elect new officers for the coming year.

The president, Jim Nielsen stated that he could not run due to other commitments, but he would like to remain as club librarian. The following nominations were made from the floor:

President	Phil Eastman
	Kim Woodward
Vice President	Nick Cirillo
	Joe Fung
	Phil Eastman
Secretary	Gerald Eskelund

Other positions can be appointed after the new board of officers is elected.

Jim Nielsen plans to hold a programming workshop for disk owners on October 6, 1979 at his home. If it works well, future workshops may be held for tape users. The purpose is to swap programs and build the club library. It is also planned as a working session.

The Secretary announced that we will attempt to collect dues for the next six (6) months beginning at the next meeting. The cost for the 6 months will be $6.00. If persons are not paid up by the last meeting in October they will be dropped from the mailing list.

The program was a continuation of the machine language programming course started by Mr. Kim Woodward. He reviewed the last session and then presented programs to add and subtract explaining the function of each step. Next lesson he plans to demonstrate a program to dump memory to tape a page at a time with a stop between pages until the space bar is pressed. There are several other functions which will also be built in to demonstrate programming principles. Bring your Red Book to the next session to be held October 25 at Computerland.

Gerald R. Eskelund

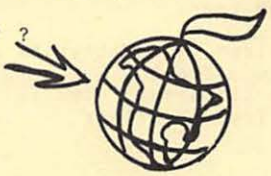
EDITORIAL

To the Officers of Apple Pi and Phil Eastman
from Bernie Urban

Thank you for your vote of confidence. It is very gratifying to me to have been chosen to represent the APPLE users' groups of the greater Washington and Baltimore areas at the October 27 and 28 meeting in San Francisco. I shall do my utmost to justify that vote. Officials from selected user groups throughout the U.S. will be attending this meeting as guests of APPLE Computer, Inc. They will be charged with the preparation of an agenda for a forum on the creation of a National APPLE Users' Group, to be held in March 1980 at the West Coast Computer Faire. Would that our budget would justify underwriting the expenses of one, perhaps two, others ---!

Still, I need your help and I need the help of our collective membership. To properly represent you at this highly significant meeting, I need to know from as many of you as possible what you consider to be the appropriate form of such a national organization, what you the users can expect from it, the merits of incorporation at a national level, who should run it and how it should relate to computer vendors and software/hardware firms, etc. Please write me at 6205 Walhonding Road, Bethesda, Md. 20016, or call 229-3458. Please help.

Next ? ?



NIBBLES

"Hands-On" Microcomputer Workbook APPLE II Edition

There is a new release from Sterling Swift Publishing Company that may be of interest to some of you. It's a 136-page paperback that when used in conjunction with four cassette tapes will teach you BASIC in a microcomputer assisted instructional mode. The paperback at $3.95 plus cassettes at $10.00 each (disks also available) seems like a convenient, fairly inexpensive way to learn BASIC at your own pace. Perhaps we can afford to purchase a set for use by our newer members on a loaner basis at minimal cost to them. Their address is P.O. Box 188, Manchaca, Texas 78652; phone (512)444-7570. They also sell "Computers and Mathematics" 454 pages, 1979, at $23.95.

At Catholic University:
The Washington Amateur Computer Society has an electronic journal which can be accessed by dialing 635-5710 (110 baud) or 635-5730 (300 baud). Type a carriage return to set rates then type HELP WACS to get the journal. A Control-O (Alpha "O") will set you back into the system. Control-S will stop the printout and Control-Q will restart. They have an interesting article on adding floppy disks and controllers and also some information on group purchases of Selectric typewriters. WACS meets the last Friday of each month. Their schedule so far is October 26, November 29 and December 28. Meeting place is Keane Hall, 1st Floor Lecture Hall at 7:30 pm until they "can't keep their eyes open".

The Fourth Annual Tidewater Hamfest-Computer Show-Flea Market will be held in the Norfolk, VA Cultural and Conventional Center October 20 and 21, 1979. 60,000 square feet of air-conditioned exhibit and Flea Market tailgating space are available. Doors open at 9:00 am. If you are into parts and junk and professional people selling amazing things contact TRC, PO Box 7101, Portsmouth, VA 23707.

We've heard about two new magazines which you may want to check out. Compute, the Journal for Progressive Computing, 900 Spring Garden Street, Greensboro, NC 27403. $9 bi-monthly. And Softside (not so new), a BASIC software magazine is up to issue 13. PO Box 68, Millford, NH 03055. The same company apparently puts out Appleseed.

Personal Software has or will be shortly releasing a new Macro Assembler. 592 Weddel Drive, Sunnyvale, CA 94086 (408) 745-7841 for dealer nearest you.

Two new stores have opened in the area. They are: Your Own Computer Ltd., 10678 Campus Way South, Largo, MD 20870. (301) 350-6680 and Program Store, 4200 Wisconsin Avenue, NW, Washington, DC 20007 (202) 337-4691.

EVENT QUEUE

NOVAPPLE October 10 at Computers Plus in Franconia and October 25 at Computerland at Tysons Corner - both at 7:30 pm. Washington Apple Pi on October 27 at George Washington University corner 23rd and H Streets NW in Tompkins Hall School of Engineering room 206 at 9:30 am.

How to Make an Integer BASIC Program Erase Itself

by Bruce F. Field

Reading Sandy Greenfarb's excellent article on integer BASIC in the September Apple Pi newsletter reminded me of a handy programming trick that some of you may not be familiar with. When booting up the Disk Operating System, a greeting program (usually written in integer BASIC) is loaded and run. This is usually a small innocuous program to print out the DOS version number, the date the diskette was initialized, etc. Once this information is printed, it would be nice if the program would disappear (i.e. delete itself). Otherwise, if we type or EXEC in a new program without first typing NEW, the greeting program will still be there to haunt us. Loading in a program, of course, will erase the greeting program.

If you have not already tried to do so, you will find that you can't type in the command NEW preceeded by a line number. Integer BASIC returns a SYNTAX ERR for this. However, a curious situation exists such that if you could somehow get the NEW command stored in the program, integer BASIC will execute it properly.

Those of you who are familiar with how integer BASIC works (or who have read Sandy's article) will understand the following procedure. If not, just follow the instructions and don't worry about how it works.

What we want to do is modify our regular greeting program so that the next to the last line is the NEW command. The last line must be an END as usual. To do this we are going to locate the end of the program at a known address in memory, and then modify it by poking the token for NEW into the next to the last line.

1. type: HIMEM:4000 (carriage return)
   NEW (carriage return)

2. load or type in your greeting program. For example:

```
10 PRINT "DOS 3.2  23 SEP 79"
20 END
```

3. before the last line, insert a new line with only a REM statement on it. For example:

```
15 REM
```

   The line number is not important, but this line must immediately preceed the line containing the END statement. Also, it is important not to type any characters or spaces after the REM statement.

To check to see if we have done everything correctly so far, type:

```
PRINT PEEK(3993)  (carriage return)
```

The result should be 93, the token for REM.

4. type POKE 3993,11

   This replaces the REM with NEW.

Now when you list the program it will have the NEW command in it. But, before you run the program be sure to SAVE it! Then you may run the program and try to LIST it; it should be gone.

Of course, this technique can be used with any program and other commands, such as CLR, can be POKEd into the program.

[Ed. note: this can be used to set LOMEM at the beginning of a program too. Line 0 should be PRINT n where n is equal to the LOMEM setting. By then PEEKing at the Program Pointer (the beginning of the program) to find its beginning and then PEEKing at the first few characters to locate the token for PRINT, you can then POKE the token for LOMEM instead. It works like a charm. Remember not to DIMension any variables prior to the LOMEM command.]

Classifieds

HELP WANTED - Need 6502 programmer for contract services short term. Work involves preparation of machine language subroutines for Apple Disk I/O. Contact Bill Barker, PO Box 36, Columbia, MD 21045 - call (301) 997-9610.

WANTED - articles for the Washington Apple Pi newsletter. No experience needed. No application required; no formal training necessary. Person should have imagination and desire to make a spectacle of him(her)self on these pages. No topic too mundane to publish. Prudence is suggested. Send to PO Box 34511, Washington, DC 20034. ^ specifically not

Classified ads accepted from members 50 words or less at no charge provided the material is obviously non-commercial. Submit your classified at least 30 days in advance attention CLASSIFIED ADS, PO Box 34511, Washington, DC 20034.

Product Reviews

MAILING LIST BY SYSTEMS DESIGN LAB review by Lee Hausman

I have been nosing around through various mailing list programs for some time now, rubbing my Susan B. Anthony's together between my fingers, reluctant to let go of them until I found a program that would really do it all, and not bankrupt me in the process. For my money, the Mailing List from Systems Design Lab is very good.

For those of you who hate to read all the way through articles like this only to find at the end that the program requires a dark blue 14.2K RAM Apple II manufactured on March 17th, let me deal with the hardware requirements right now: you will need a 32K RAM minimum Apple (your choice of colors), an Applesoft ROM Card, a disk system and, obviously, a printer.

To get started, you may enter data and SAVE it as a file for later use. A total of 120 names per file may be used for 32K systems, 300 names per file for 48K systems. By switching files around, you could work with over 20,000 names! (This by virtue of the fact that a disk will only recognize a total of 84 file names). Data fields supported include two lines for names, street address, city-state-ZIP, and a code of your choice. The code is quite handy, as it is usable for Sort and Search functions, and prints on a listing of a file, but doesn't print on the mailing labels. Various functions allow you to work with your data either on your video screen or via your printer.

Commands available include:

- LOAD from disk
- SAVE to disk
- ADD records to file
- CHANGE records in file
- DELETE records in file
- SORT on any field
- PRINT multiple labels of one record
  (handy for your return address!)
- PRINT entire file on paper (not label format)
- PRINT labels
- SEARCH and PRINT labels
- LIST entire file on video
- SEARCH and LIST entire file on video

Three additional commands allow you to set print format parameters by setting spacing between labels, left margin, and printer slot number. And printing or listing by printer or video screen can be halted or interrupted and restarted with a few key strokes.

The program is distributed on cassette and is easily LOADed and SAVEd to disk. And it is in Source Code so that you can easily make modifications for your needs. I added a special printer patch for my notorious Serial Interface Card. And adequate instructions come with the program. Perhaps best of all, this program will only set you back as much as a roll of 15¢ stamps! Just $15.00 from Systems Design Labs, 2612 Artesia Blvd, Suite B, Redondo Beach, CA 90278.

[Ed. Note: lest you be possibly mislead by Mr. Hausman's statement concerning the 20,000 names he mentions you can work with, bear in mind that a single side of a disk only permits about 101,000 characters of information which (by my trusty calculator) works out to about 1,200 78 character records. Check with Systems Design Labs to be sure.]

IDS MODEL 440 "PAPER TIGER" PRINTER review by Fred P. Sharp

Practically every home computer enthusiast dreams of acquiring a versatile, dependable and economical printer for his system. After hearing about the IDS Model 440 at the Washington Apple Pi Users meeting, I was convinced that this was going to be the printer for me. Although I was convinced the Paper Tiger was for me, my bank account kept telling me it wasn't. About that time my employer provided me with sufficient funds to purchase the printer to work with an Apple II Plus Computer in the office. After the printer with serial interface arrived and the initial minor interfacing problems were taken care of, I was convinced this printer meets at least two of my original prerequisites, versatility and dependability. The third requirement, economy, I am afraid may never be met by any printer, that is on my budget. Enough editorializing; now to the review.

The printer offers four different print sizes: 8.3, 10, 12.5 and a very small 16.5 characters per inch. In addition to the four normal print sizes there is an enhanced version for each. See the samples below. The print size and type (normal or enhanced) is controlled by either direct mode selection or control codes embedded in your program. One problem I have encountered is slowing the output down so the printer can keep up. Although the Apple's Serial Interface card allows for a delay at the end of each line, it has proven insufficient for the task even with the baud rate set at the maximum 1200 bits per second allowed by the printer, I have reverted to the BASIC SPEED command (SPEED = 190). Although this also slows down the output to the screen, if the commands are properly placed, the speed can be reduced

to 190 during the print routine and restored to 255 for CRT display later. One reason for this is the limited buffer (256 bytes) on the non-graphic model. The graphic model comes with 2048 bytes of buffer. With that added option the problem might evaporate. On the positive side (and there are many) the Paper Tiger lets you format your printout to accommodate almost any kind of form. Print 6 or 8 lines per inch. Choose any line size up to 132 columns across. Print whatever size you need, from address labels to legal-size reports. The Paper Tiger has eight switch-selectable form lengths. You can adjust the tractor width from 1.75 to 9.5 inches. At the flip of a switch, this machine skips over form perforations automatically. Aligning forms is also simple. Put the Paper Tiger offline, and incrementally move the form up or down with the touch of a switch. If you are worried about special paper, not with this printer. It prints on plain, ordinary paper. If you need extra copies, the Model 440 can print on multiple copy forms. When the paper supply, whether roll or fan fold, runs low, an out-of-paper detector automatically stops the printer, places it offline, and lights an indicator for the operator.

Now that I have access to a printer, I am excited abut the new Word Processor package being introduced by MUSE for $99.00 at the Philadelphia Computer Conference. I have promised Bernie Urban a Software review on this new package in the future.

THIS IS AN EXAMPLD OF 16.5 CPI IN THE NORMAL MODE.

THIS IS AN EXAMPLE OF THE 12 CPI IN THE NORMAL MODE.

THIS IS AN EXAMPLE OF THE 10 CPI IN THE NORMAL MODE.

THIS IS AN EXAMPLE OF THE 8.3 CPI IN THE NORMAL MODE.

THIS IS AN EXAMPLE OF THE ENHANCED 16.5 CPI.

THIS IS AN EXAMPLE OF THE ENHANCED 12 CPI.

THIS IS AN EXAMPLE OF THE ENHANCED 10 CPI.

THIS IS AN EXAMPLE OF THE ENHANCED 8.3 CPI.

8.3 CHARACTERS PER INCH, **AND ENHANCED.**

10 CHARACTERS PER INCH, **AND ENHANCED.**

12 CHARACTERS PER INCH, **AND ENHANCED.**

16.5 CHARACTERS PER INCH, **AND ENHANCED.**

BUFFER FULL TEST
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←`abcdefghijklmnop
qrstuvwxyz{|}~ !"#$%&'()*+,-./0123456789:;<=>?@ABCDE
FGHIJKLMNOPQRSTUVWXYZ[\]↑←`abcdefghijklmnopqr
stuvwxyz{|}~ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVW

Multiprocessing with the APPLE

by Bruce F. Field

The eternal cry of non-computer freaks is "What does it do?" or "What good is it?". After we end up making lame excuses (instead of admitting we wanted it because it's a neat toy), we realize that there actually are some household uses for a computer in monitoring and/or controlling household utilities. For example, we could use the computer to monitor the weather conditions outside and heat load inside the house to optimize the heating or cooling rate to minimize energy consumption. Monitoring freezer temperatures, dampness in a besement, or energy usage of individual electrical appliances are simple tasks for a computer, and a computer system also includes the capability of making intelligent decisions, sorting out false alarms, etc.

My particular interest is in monitoring the weather. The computer can answer questions such as "Has it rained enough or do I have to water the garden?", or "Should I put on a coat before I go out?". Sure I can look at the thermometer and see that it's 50°F out, but the thermometer doesn't tell me that the wind is blowing at 30 mph and the effective wind chill temperature is 28°F, whereas the computer can.

Now that we've demonstrated some semblance of usefulness for a computer, how can we use the Apple to perform some of these monitor and control functions? Unfortunately, the Apple is not very well suited for this kind of work. All of the examples mentioned above and almost any other control type problem I can think of require 24 hour/day operation. What happens when your kid wants to play a game, or you want to balance your checkbook? You have to stop the monitor program! Depending on the application, it may be okay to temporarily stop the program, save all the variables and restart it later, but many times the program cannot be stopped and in any case it is inconvenient. Here is where hardware oriented and software oriented people diverge in their thinking. One solution is to develop a software multasking operating system so that two programs can be run on the Apple "simultaneously". (In reality the programs are executed sequentially, it only appears to the user that they are executing simultaneously.) If the two tasks are to be completely independent from one another this method requires writing some fairly sophisticated machine language software including rewriting some of the existing Apple Monitor software and replacing the original ROMs in the Apple. A second solution is to use another computer. Before you say "this is crazy - why should I buy another computer", think of this: small dedicated microcomputers containing a CPU and a few support chips are CHEAP. If we use the Apple as a master computer (or terminal) the small slave computer only needs a CPU, a small PROM or ROM, maybe 1K bytes of RAM, an A/D converter with several analog input channels, and a few I/O lines.

In this configuration, programs for the slave are stored using the mass storage of the Apple (disk or cassette) and downloaded into the slave RAM. The slave PROM must contain at a minimum, software to communicate serially with the Apple, and a program to load memory and start an application program. If desired, application programs could also be stored in the PROM, but I prefer to put the programs in RAM as it makes it easier to write and debug them. Also, the function of the slave computer can be changed almost instantly by downloading a new program from the Apple.

Advantages of using a slave processor are that it completely frees the Apple for other uses, additional expensive peripherals (video display, mass storage) are not needed, and the slave system consumes much less power than the Apple - it is economical to let it run continuously. The only serious disadvantage (other than cost) is that application programs must be written in machine language. Putting a BASIC interpreter in the slave significantly increases the amount of memory required (and as far as I am concerned, costs too much). However, if someone has a compiler for BASIC I am interested. The program could then be written in BASIC, compiled on the Apple, and only the machine language code and run time library would be downloaded to the slave.

This brings up the next point - cost. A small system if assembled from components should cost no more than $150 and considerably less if one has a well stocked parts box. A rough schematic of the slave system I have built is shown in the figure. Many details have been omitted for clarity and because of laziness. It is constructed on a perfboard 4¼"x 7½"with wire wrap sockets. The perfboard forms the lid of a plastic experimenter's box (2 3/8" deep) with the power supply contained within the box. Because of the low power consumption no fan is needed. If I remember correctly I had almost all the parts already on hand. This explains the apparently curious choice of a SC/MP processor (by National Semiconductor) as the CPU. It turns out however that this is actually a reasonable choice as the SC/MP was designed for control applications and is very simple to interface and program. National Seminconductor at one time sold (and may still sell) a board containing a SC/MP, 256 bytes of RAM, and a 512 byte ROM to communicate to a teletype, the whole thing costing $99. This could be used directly with almost no modification, except for the addition of a A/D converter and maybe some more RAM memory.

To minimize parts count and cost, no UART or handshaking is used for the serial communcations between the processors. Instead, all parallel to serial conversion is done in software with careful timing. This prohibits half duplex operation (where the slave echos all transmitted data back to the Apple) but with a short direct connection between the two machines half duplex is not necessary. Thus, only three connections are required between the slave and the Apple: all three being connected to the Apple game I/O connector.

AN$\emptyset$ - outputs data from the Apple to the slave

SW$\emptyset$ - inputs data to the Apple from the slave

Gnd - a common ground connection

The slave has it's own separate power supply so that it's ground must be connected to the Apple ground for proper operation.

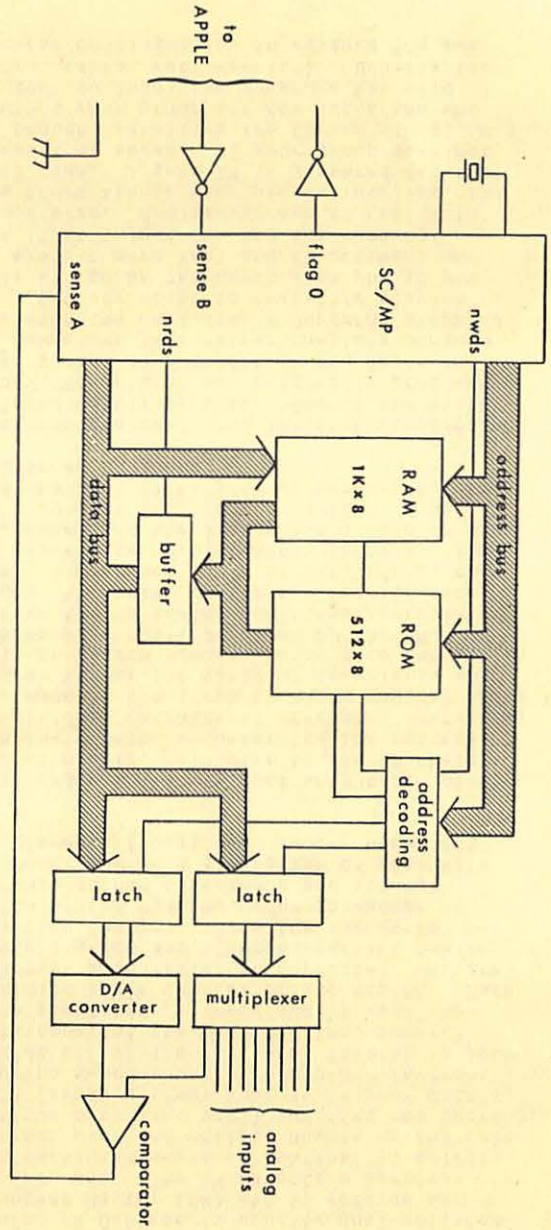
The 512 byte ROM contains a serial communications program and a monitor program to load memory, dump memory, and start a program loaded in memory, all under control of the Apple. The D/A converter is used for analog-to-digital conversion of eight analog inputs which are selected one at a time by the multiplexer and compared to the output of the D/A converter. A successive approximation technique is used to adjust the D/A output allowing fast conversion (approx. 200 µsec) even with the slow SC/MP. Using a D/A converter and a comparator is cheaper than an A/D converter and the D/A can also be used to provide a programmed voltage if necessary.

Thus we now have all the required elements to periodically monitor eight different parameters, store the data in memory, and at infrequent intervals under operator control, dump the data to the Apple for analysis and storage. If external devices are to be controlled an output port can easily be added.

In conclusion, a word about using other processors for the slave. If you are starting from scratch, it seems reasonable to choose a 6502 (or equivalent) processor for the slave. This will considerably simplify writing application programs as they can easily be tested on the Apple (also assembled if you have an Assembler). A KIM can be used with some modification, although it really is overkill. I am not too familiar with the TIM (KIM's cousin) but I think it should be perfect for this application, if there are any more TIMs around.

I have tried to avoid too much detail but still explain one way of connecting a second processor to the Apple to continuously monitor eight analog signals. If anyone has any suggestions, questions, etc., drop me a note at the address below.

Bruce F. Field
1402 Grandin Avenue
Rockville, MD 20851



On the Nature of Survival:

Simulation by Mark Crosby

Games have long held a particularly important place in the evolution and development of *homo sapiens.* Throughout our history we have been in competition with each other not only for survival but for sheer diversion. We are all familiar with the day-to-day struggle to survive. Although you might not often think of it, the daily struggle is very real and even a simple error in judgement can mean starvation, personal loss or even death. Modern Man has all but eliminated physical combat in day-to-day encounters in favor of intellectual "combat". While this seems to be a favorable situation, the dearth of emotional outlets in day-to-day living quickly builds up. Our emotional requirements demand that we have an excess of stimulating challenges to relieve pent-up emotions. Enter Dungeons and Dragons.

This simulation - which is mistakenly called a "game" is really a substitute physical challenge for the human survival instinct. When properly presented, it can "fool" us into fantasizing we are <u>really</u> fighting a dragon or a warrior that requires our quick thinking and memory of past experiences to "win". It can be a substitute for the "blowing off steam" we all require and, thus, is good for our mental health in the long run. Since young people do not always have the opportunity of testing their reactions and survival techniques directly, a Dungeons and Dragons simulation can actually help prepare one for later, and more demanding, challenges that we encounter during our lives.

Dungeons and Dragons (D&D) relies heavily on "physical" combat, instinct and ingenuity in its execution which is an emotional substitute for Man's earlier bloody history (man slaying animal for food, etc.). When is the last time you had to (yourself) kill another person to protect yourself or kill an animal for that matter, or even defend a source of food from pillaging? The self-satisfaction that "I can survive" is psychologically soothing to us. In a very real way, then, a well-presented Dungeon can help us survive daily tensions and challenges in real-life situations.

Acting out fantasies during a Dungeon can release all sorts of emotions in people who, formerly, were stoic. Angered and backed into a corner during a Dungeon, people display genuine insight and come up with ingenious solutions to particular problems. They squabble, barter, cheat, yell out, suffer the agony of defeat, and laugh. Then with a cry of triumph and a release of tension the participant beams at the success of winning a campaign.

Without going into the many details of exactly what is a D&D simulation I will say this: A Dungeon is usually goal-oriented and administered by a Dungeon Master (DM) who is referee and all opponents rolled up into one. The DM selects a particular goal ("Find the lost enchanted sword of Ashtibal"), selects the terrain ("You will cross over the Hdilets Mountains and come to a house and will continue over land until you find the sword"), and selects opponents ("A large, vicious bear is in your path"), random events ("A shimmering sword appears in mid-air in front of you"), etc. You provide all of the decisions leading to your hopefully successful attainment of the goal. Often, several people participate in the simulation at once and "travel" together in a group to maximize their chances of succeeding. Each of these participants chooses a character or characters who are named, have strength, intelligence and other human-like qualities who will "do" the action for you. Upon the successful (or non-successful) completion of the Dungeon, your character is either dead, depleted or has gained experience and strength. A Dungeon, then, can be described as a simulation of real life with additional fantasy elements (sometimes) and is often set in medieval times.

There is much to do in a Dungeon: The DM must keep track of the whereabouts of the group of people, roll dice to see if "Fate" puts a vicious animal in their path, or determine the outcome of a fight (depending on individual characters' strength, dexterity, etc.), referee decisions made by the group, or allow magical items to be found which might aid or hinder the party of travelers, etc. Much of it relies heavily on random numbers using dice and tables of outcomes which have been previously prepared by the DM. A single character might have the following qualities represented by a number or percentage: Strength, dexterity, intelligence, charisma, armour, weapons, appearance (ugly or beautiful), race (human or other), luck, experience, useful items (knives, flint set, water bottle, food, rope) and the list doesn't have to stop there. Persuasiveness, alignment (for good or evil or somewhere in-between), mental balance (likely to go berserk?), etc., are often included in advanced Dungeons.

Existing computer simulations are small and limiting compared to Dungeons prepared and played weekly by some. One of the better known computer simulations, "Adventure" was originally programmed on a larger computer then scaled down to fit in the Apple and other micro-computers. Adventure is a rather complete Dungeon but it not as complex as even the most simple Dungeons prepared for real-life execution. One advantage to real-life playing is that you have real people who go on the quest with you so you have real arguments and exhibit more real and (sometimes) uncontrolled emotions than if it's just you and the computer finding your way through a maze. A disadvantage to real-life playing is that, because there are so many people involved, the simulation can get bogged down. A good DM is prepared for that though by adding the element of speed. If your group does not reach decisions quickly enough, penalties are thrown at you, e.g., ("The dragon suddenly rushes your group and now let's see who survives" or "Since you took so long, the treasure has slid down into the abyss and you cannot retrieve it"). Usually the DM prepares for all of these contingencies in advance (An awesome task!)

One parting comment. In a recent newspaper article concerning a missing college student, it was alledged that, perhaps, it had something to do with a group of people who regularly play D&D. While it is entirely possible that REAL dungeons exist somewhere in the U.S., most D&D players prefer to do it all on paper around the dining room table with dice. Real weapons and real situations are generally disapproved of although miniature playing pieces (dragons, warriors, etc.) are often employed to help illustrate players' relative positions to each other as they travel through the Dungeon. There are stores which cater to this crowd, selling everything from the miniatures to 20-sided dice and even costumes.

In future articles, I plan to cover other aspects of simulation particularly as it applies to D&D. I hope to illustrate maze generation, random number techniques and percentiles, generating characters to use in Dungeons, group dynamics in a Dungeon, etc. I would be pleased to hear from anyone who is interested in contributing toward this effort. If enough interest is generated, perhaps we can begin a regular column. Address all inquiries to the Washington Apple Pi PO Box in care of myself.

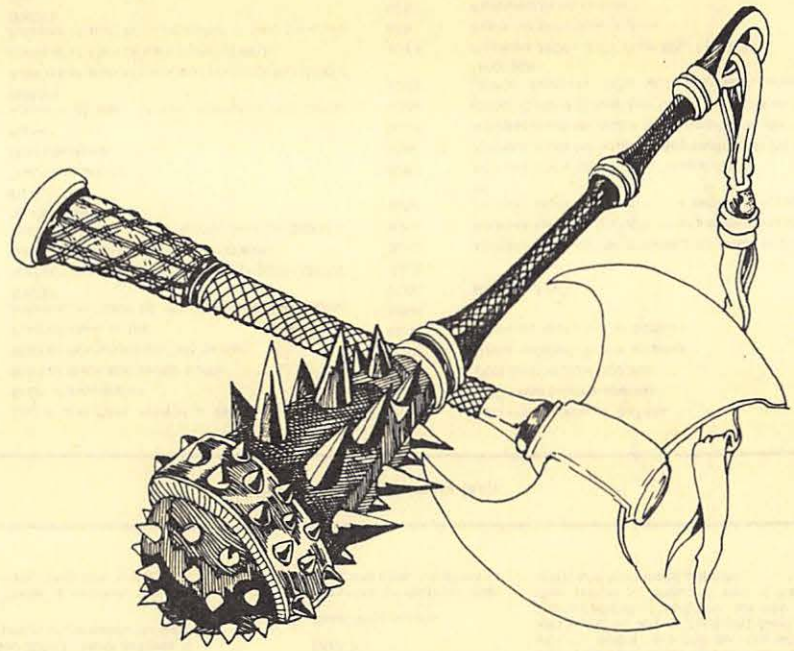
Some gaming contacts:

TSR Periodicals
PO Box 110
Lake Geneva, WI 53147
THE DRAGON

the Chaosium
PO Box 6302
Albany, CA 94706
DIFFERENT WORLDS

Game Designers' Guild
c/o Randy Reed
2803 Goodwood Road
Baltimore, MD 21214

Chimera Books (a good local store)
1776 E Jefferson
Rockville, MD 20852
(301) 770-3444



Disassembling the DOS 3.2

You "Can't tell the players without a score card" and you can not effectively use the Apple II DOS 3.2 without this important information on its organization.

William Reynolds
1733 N. Ford Street
McMinnville, OR 97128

On the surface, DOS 3.2 is identical to DOS 3.1. Upon booting, the DOS is loaded (slave or master), the greetings program is run, MAXFILES defaults to 3, and HIMEM is set at $9600. DOS 3.2 still communicates with the rest of the APPLE via input and output hooks at $36, 37, 38, and 39. (All addresses refer to a 48K machine.)

The differences are many: In Applesoft, DOS does the call 3314 or call 54514 automatically, volume checking is ignored unless explicitly defined in the command, and the system defaults to NOMON C,I,O status. The hooks at $36 and 37 (the print routine) now contain $9E81. The routine to restore DOS is now at $9DBF. This can be called if page 3 is overwritten. The command and error message tables are in different locations. The command table is the same as in the DOS 3.1. The error messages, however, are quite different. After a BLOAD, A$ is now found at $AA72,3; L$ is now found at $AA60,1.

When the keyboard input routine (9E81), is called, DOS checks the mode. If it is in direct mode, the DOS reads the keyboard, then goes to the print routine. The print routine has seven routines of it's own, 0-6. It calls the correct one, depending on whether the mode is direct, deferred, execute, read or write, etc. These routines are all inter-related.

In direct mode, when a return is detected, DOS attempts to match the string in the keyboard input buffer ($200-2FF) to a command in the table. In the print mode, direct or deferred, it stores all characters in the keyboard input buffer until a return is detected. It then checks for a CTRL-D as the first character. If not found, DOS drops out and returns control to wherever it came from. However, if Control D is detected, DOS attempts to match the string to the command table. If a match is not made, it prints "Syntax Error".

When DOS matches a command, it then checks for names, if needed, or numbers, if needed. After getting all data required, a check for optional data is made. After any optional data is read, numbers are changed to hex if need be, the maximum and minimum ranges are compared, then if all data is OK, the number is stored and DOS returns to check for any other optional data.

A routine gets the correct address from the stack, then executes the command. I have highlighted a few of the commands:

PR# and **IN#** do the same function as in BASIC, except that DOS will set the hooks properly before releasing control.

MON and **NOMON** set a mask at $AA74 as follows: 0 = monitor nothing, $10 = monitor 0, $20 = monitor I, $40 = C, and combinations thereof.

MAXFILES resets HIMEM and PP (INT BASIC) and allocates a file buffer via a subroutine at $A7D4.

BRUN does a BLOAD then a JMP ($AA72).

RUN does a load, then jumps to a routine which executes the program.

Which routine is used is dependent upon which language is being used, BASIC, FP RAM, or FP ROM.

LOAD reads the file type and does either INT or FP as needed, then loads the program. When in FP mode, and after the program is loaded, DOS does the call 3314 or call 54514 as needed to set the program pointers for Applesoft.

FP attempts to find a ROM card and turn it on. If possible, it sets the return addresses via a routine at $9D84. If no card is found, the DOS runs Applesoft, then goes to a routine at $9DEA to set return addresses correctly.

INT makes certain the ROM card is off, then goes to $9D84 to set return addresses correctly.

If a person wishes to use DOS from a language or operating system not standard to the APPLE, there is no problem, unless an error is detected. If you do not wish an error message to cause a return to BASIC or Applesoft, the address at $9D5E and F can be changed for your particular system.

Whenever a change in language is done, DOS updates it's return address stack from the stack for that particular language. All commands except PR#, IN#, MON, NOMON, INT, FP (if in ROM), and MAXFILES go through routines that use file buffers.

All commands may be called from monitor or machine language, provided (1) A language change is not needed, (2) the file names have been placed into the name buffer(s), and (3) that any other parameters have been properly placed into their locations as needed.

The disk controller card contains two (2) PROM's, 256 bytes each. One PROM contains the program to start the booting of the DOS. The other is used for a program that, together with some other IC's, actually controls the head position, reading a bit, writing a bit, sending the byte to the APPLE bus, and getting a byte from the APPLE bus. The following locations control the hardware functions. Add 00S0 to each address, S = the slot number of the controller card.

C080-87 These addresses sequentially step the motor that moves the head back and forth. Odd addresses step one way, and even addresses step the other way.

C088 Turns off the drive motor.

C089 Turns on the drive motor.

C08A Enables drive two.

C08B Enables drive one.

C08C,D Control connecting the APPLE bus to the hardware for strobing the byte in or out of the 74LS323 IC shift register, depending upon the previously set status of C08E,F.

C08E,F Read/Write control.

I have documented all routines, subroutines, buffers, and other locations to the best of my ability in the memory maps that follow. Notes tell the function and usage of each. On most items I have given only the starting address. The end address is implied to be the next documented location minus one. On stacks of addresses, the parenthesized number is the number of addresses contained in that stack. Remember that any two-byte items are always stored low byte first. Documentation of addresses in the B000-BFFF area may be in error because that area got too complex for me to retain my sanity.

My thanks to my family for their time and patience, to other persons for their articles on DOS functions, APPLE for their excellent documentation, without which I would have had no idea what was going on, and to Terry and Kent at Computerland of Portland, for use of their printer to obtain 60 feet of hard copy, and their moral support.

APPLE II DOS 3.2 Memory Map

95FF	End of user RAM: HIMEM = 49151
9600	Start of data buffer
9700	Start of track and sector buffer
9800	Start of miscellaneous info buffer
982D	Start of name of file
984B,C	Address of start of miscellaneous info buffer ($9800)
984D,E	Address of start of track and sector buffer ($9700)
984F,0	Address of start of data buffer ($9600)
9851,2	Address of start of name buffer, next file ($0000 = no more files)
9853	Data
9953	Track and sector
9A53	Miscellaneous
9A80	Name
9A9E,F	Address of start of miscellaneous info buffer ($9A53)
9AA0,1	Address of start of track and sector buffer ($9953)
9AA2,3	Address of start of data buffer ($9853)
9AA4,5	Address of start of name buffer of next file down ($982D)
9AA6	Data
9BA6	Track and sector
9CA6	Miscellaneous
9CD3	Name
9CF1,2	Address of start of miscellaneous info buffer ($9CA6)
9CF3,4	Address of start of track and sector buffer ($9BA6)
9CF5,6	Address of start of data buffer ($9AA6)
9CF7,8	Address of start of name buffer of next file down ($9A80)
9CF9 – 9CFF	Unused
9D00	Address of name of first file
9D02	DOS keyin routine address
9D04	DOS print routine address
9D06	Name number 1 buffer address
9D08	Name number 2 buffer address
9D0A	
9D0C	Bottom of DOS
9D0E	
9D10	Address stack for the internal print routines (7)
9D1E	Address stack for the DOS command routines (28)
9D56	Address stack for return to the current language (6)
9D62	Address stack for return to Integer BASIC
9D6C	Address stack for return to Applesoft ROM (6)
9D78	Address stack for return to Applesoft Disk (6)
9D84	(3D3G) Control B, re-enters INT or FP (ROM only)
9DBF	(3D0G) Restores DOS and re-enters current language
9DEA	Restores $3D0 – $3FF from $9E51 – $9E80
9E51	Stack for the above routine
9E81	Keyboard input routine
9EBD	Calls correct internal print routine, depending upon mode
9ED1	Restores keyboard and print hooks
9EEB	Internal routine for information from the disk
9F12	Internal routine for printing
9F23	Prints and exits DOS
9F2F	Keyboard input internal routine
9F52	Internal routine for sending information to disk
9F61	Routine to correct internal routine
9F71	Used by the EXEC command
9F83	Mask MON status, print and exit

9FC8	Does a RETURN
9FCD	Start of section that attempts to match to a command and get all information needed and all optional information given. Checks syntax and ranges before execution.
A229	PR# routine
A22E	IN# routine
A233	MON routine
A23D	NOMON routine
A251	MAXFILES routine
A263	Start of DELETE routine
A271	Start of LOCK routine
A275	Start of UNLOCK routine
A27D	Start of VERIFY routine
A281	Start of RENAME routine
A298	Start of APPEND routine
A2A3	Start of OPEN routine
A2EA	Start of CLOSE routine
A331	BSAVE routine
A35D	BLOAD routine
A38E	BRUN routine
A397	SAVE routine
A413	LOAD routine
A4D1	Run routine
A4E5	Runs Integer BASIC program
A4F0	CHAIN routine
A4FC	Runs FP ROM program
A506	Runs FP RAM program
A510	WRITE routine (set up)
A51B	Read routine (set up)
A54F	INIT routine
A56E	Catalog routine
A57A	FP routine
A59E	INT routine
A5C6	EXEC routine
A5DD	Position routine
A60E	Starts the read process
A626	Starts the write process
A644	Stores data coming from text file into keyboard buffer. Used by the EXEC command.
A65E	Error checking?
A679	Closes files, exits DOS
A682	Goes to hardware routines
A69D	Sets up address of name section of next file
A6AB	Close the buffer last used
A6C4	Prints, "SYNTAX ERROR "
A6C8	Prints, "NO BUFFERS AVAILABLE"
A6CC	Prints, "PROGRAM TOO LARGE"
A6D0	Prints, "FILE TYPE MISMATCH"
A6D5	Prints other error messages by message number contained in $AA5C
A71A	Moves parameters given to locations for use by hardware routines
A743	Moves name from the name buffer to the name section of the file buffer
A74E	Moves addresses of sections of file buffers to locations for use by hardware routines
A764	Attempts to find a file buffer already in use by the name given
A74F	
A7C4	Checks file type
A7D4	Sets up file buffers and addresses (used by MAXFILES)
A851	Restores DOS hooks
A884	Start of command table
A909	This is a table of two-byte masks. One byte is used to determine what type of extra data is needed by a command. The other byte is used by the hardware routines for what file type to create or look for.
A941	Table containing the letters V, D, S, L, R, B, A, C, I, O. This is used when checking for optional data.
A94A	Table of bytes for determining what type of optional data to look for.
A995	Table of minimum and maximum ranges for V, D, S, L, R, B, A.
A971	Start of error message table
AA3F	Relative address of start of error message, i.e. ($A971,X)
AA4F,50	Address of name section of next available file buffer
AA51	
AA52	Internal print routine number
AA53,4	PR# hooks out of DOS
AA55,6	IN# hooks out of DOS
AA57	Number of total file buffers
AA58	Number of file buffers not in use
AA59 –	Temporary storage used by various routines
AA5E	Mask for MON and NOMON
AA5F	Command number
AA60 – 61	Found L$ from a BLOAD
AA62 – 65	Temporary storage used by various routines
AA66,7	Defined volume number
AA68,9	Defined drive number
AA6A,B	Defined slot number
AA6C,D	Defined length
AA6E,F	Defined record number
AA70,1	Defined byte number
AA72,3	Defined address
AA74	
AA75	Start of file name buffer number 1
AA93	Start of file name buffer number 2
AAB1	
AAB2	Control D
AAB3	Mode (direct, deferred, etc.)
AAB4,5	
AAB6	Value used for language, e.g. INT = 0, FP RAM = C0, FP ROM = 80
AAB7	
AAB8	The name, "Applesoft"
AAC1,2	Address of start of IOB (used by RWTS)
AAC3,4	Address of start of buffer for track/sector list (used by RWTS)
AAC5,6	Address of start of buffer for data (used by RWTS)
AAC7,8	Top of total RAM in the APPLE II
AAC9	Address stack for hardware routines (14)
AAD5	Address stack for hardware routines (6)

AAF1	Address stack for hardware routines (6)
AAFD	Goes to the correct hardware routine
AB28	Reads VTOC and reads directory attempting to find an entry with the same name as the one given. If not found, checks the table of masks to see if it is allowed to create a file. If it may, it does so, and if not, it exits with "FILE NOT FOUND" or "LANGUAGE NOT AVAILABLE"
ABDC	Clears miscellaneous info hardware buffer; sets volume number, drive number and slot number.
AC06	Close routine. Updates VTOC, track bit map, and sector count of directory entry as needed.
AC3A	Rename routine. Finds directory entry, stores new name in entry, then writes that directory sector back to disk.
AC58	Goes to correct hardware routine
AC70	Goes to correct hardware routine
AC87	Sets parameters for following routine
AC8A	Actually reads text file
AC93	Sets parameters for following routine
AC96	Reads program or binary file
ACA8	Puts byte being read into buffer
ACBB	Sets parameters for following routine
ACBE	Writes into text file
ACC7	Sets parameters for following routine
ACCA	Writes program or binary file
ACDA	Gets byte being written from buffer
ACEF	Lock hardware routine
ACF6	Unlock hardware routine
AD12	Sets parameters for following routine
AD18	Verify hardware routine
AD2B	Delete hardware routine
AD54	Part of delete routine, frees sectors used by deleted file.
AD98	Catalog hardware routine
AE42	Part of catalog, prints the number in $44 as three digit ASCII.
AE6A	Moves miscellaneous info from the file buffer to the hardware buffer.
AE7E	Moves miscellaneous info from the file buffer to the hardware buffer.
AE8E	Initialize hardware routine
AF08	Sets 42 and 43 as pointers to sections of the file buffer
AF1D	Writes data section of file buffer to disk
AF34	Writes track/sector list section of file buffer to disk
AF4B	Sets hardware pointer to the track and sector list section of the file buffer being used
AF5E	Checks position in file. If out of current sector, reads/writes next sector, updates VTOC buffer, updates track/sector list section of file buffer if in write mode.
AFDC	Reads from disk into data section of file buffer
AFE4	Sets hardware pointers to data section of file buffer being used
AFF7	Reads VTOC to its buffer ($B3BB – B4BA)
AFFB	Writes VTOC from its buffer
B011	Reads a directory sector into its buffer ($B4BB – B5BA). Initially reads sector A, successive entries into this subroutine read successive sectors from the disk. When all sectors have been read and the subroutine is called again, it will merely exit with the carry set.
B037	Writes current directory sector from buffer to disk.
B052	Sets up IOB for directory sectors, goes to RWTS
B0A0	End of above if no error
B0A1	Start of error handling routine for above
B0B6	Checks position in file, reads/writes next sector as needed
B134	Initializes data section of file buffer to all zeroes
B15B	Sets next position in file
B194	Increments position in file
B1A2	Sets next RAM address
B1B5	Calculates how much RAM is left
B1C9	Reads VTOC and successive entries, attempting to find the specified file name.
B21E	Puts name of file into directory
B224	Sets next sector, updates VTOC buffer
B2C3	Updates VTOC
B2DD	Calculates track bit map for VTOC
B300	Sets/checks parameters for file?
B35F	Routine with different entry points to exit the hardware routines with error
B397 – A6	Temporary storage for hardware routines
B3A7 – AA	T, I, A, B Used by catalog for file types
B3AB,C	
B3AD – BA	In reverse order, the string, "DISK VOLUME"
B3BB	VTOC buffer
B4BB	Directory buffer
B5BB – D0	Temporary storage for hardware routines
B5D1 – FF	Miscellaneous info section of currently used file
B600	Buffer. Purpose?
B700	Reads drive 1, current slot, $B1 sectors, track 0, sector A into RAM starting at $1B00. Boot routine?
B74A	Writes $0A sectors, starting from $B600, then $1B sectors, starting at $1B00, beginning at track 0 sector 0.
B793	Increments track/sector as needed and data address for above two routines
B7B5	Calls RWTS, checks status upon return
B7C2	Sets address of data buffer, and sets expected volume number
B7DB	Stores zeroes in one page, starting at the address in $42, 43
B7E7	Start of IOB and device characteristics table
B800	Part of RWTS?
BA90 – FF	Temporary storage for RWTS?
BB00	One-page buffer (RWTS?)
BC00	One-page buffer (RWTS?)
BD00	Start of RWTS
BFD4	End of RWTS
BFD5	Various endings sections for the hardware routines
BFFF	End of RAM

Plotpourri by Samuel S. Cottrell

The program described in this article was inspired originally by Wm. Games' article "Patterns" in the Nov-Dec 1978 issue of Creative Computing, and more recently by the cover of MICRO for September 1979. I never cease to be impressed with the Hi-Res plotting capability of the APPLE, and yet nobody seems inclined to demonstrate it very well (including Apple Computer Co.!). On the other hand, we see the crude attempts of the "other" machines filling the pages of the micro rags. Rather than just snicker, this time I thought I would offer to my fellow Pi-men (and ladies!) my feeble efforts [*Not at all. Ed.*] to exploit Apple Hi-Res in a pseudo-3D (isometric) mode.

The program is a very simple one, and plots a 3-D-like picture of ten different functions of two variables on the HGR2 screen. It will run in either tape or ROM Applesoft. You are asked initially "Which plot first?" so you can select a favorite if desired. The program will then plot all the functions in sequence, returning to the first and repeating. The program unfortunately runs at about the same speed that I type (about 1½ baud) so be patient. Those trig, log and exponential algorithms are pretty time-consuming, and each plot requires up to 17,400 evaluations of the equation being shown. In line 340 I have included an alarm clock in case you fall asleep waiting for the picture to be completed. Line 345 lets you ponder the current plot before proceeding to the next (hit any key).

For those who are interested, the functions being plotted are contained in lines 35,45,55,...125, and the controlling parameters for the plots precede the equations in lines 30,40,50,... 120. They are:

- X1,X2 — X lower and upper limits
- Y1,Y2 — Y lower and upper limits
- ZS — Z scale factor (vertical exaggeration)
- RZ — "resolution" ("high"=2, "low"=1)
- PC — plot color
- BC — background color

The "guts" of the program are in lines 200 - 330, and the rest is administrative stuff.

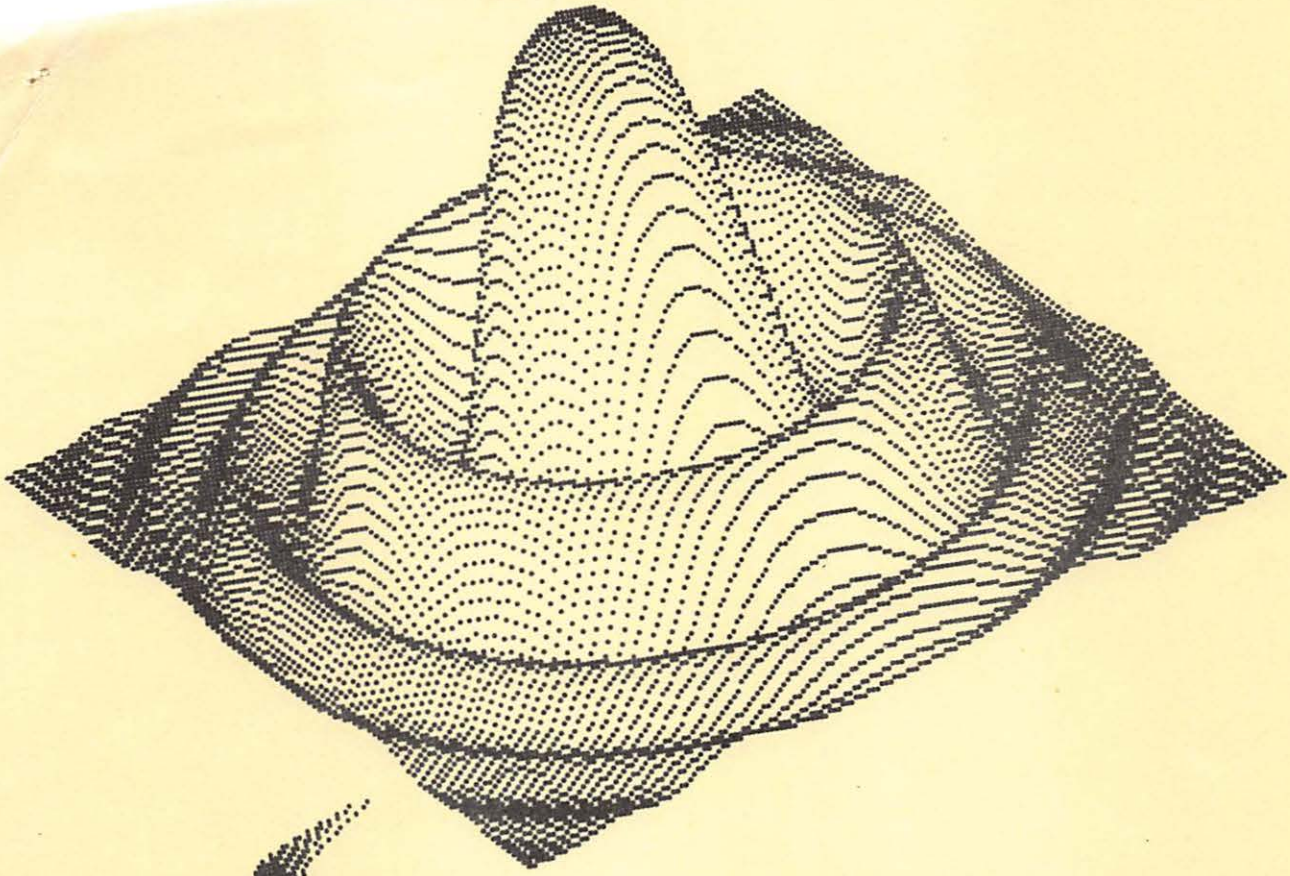
Some suggestions: Adjusting the color on your set can produce some interesting effects

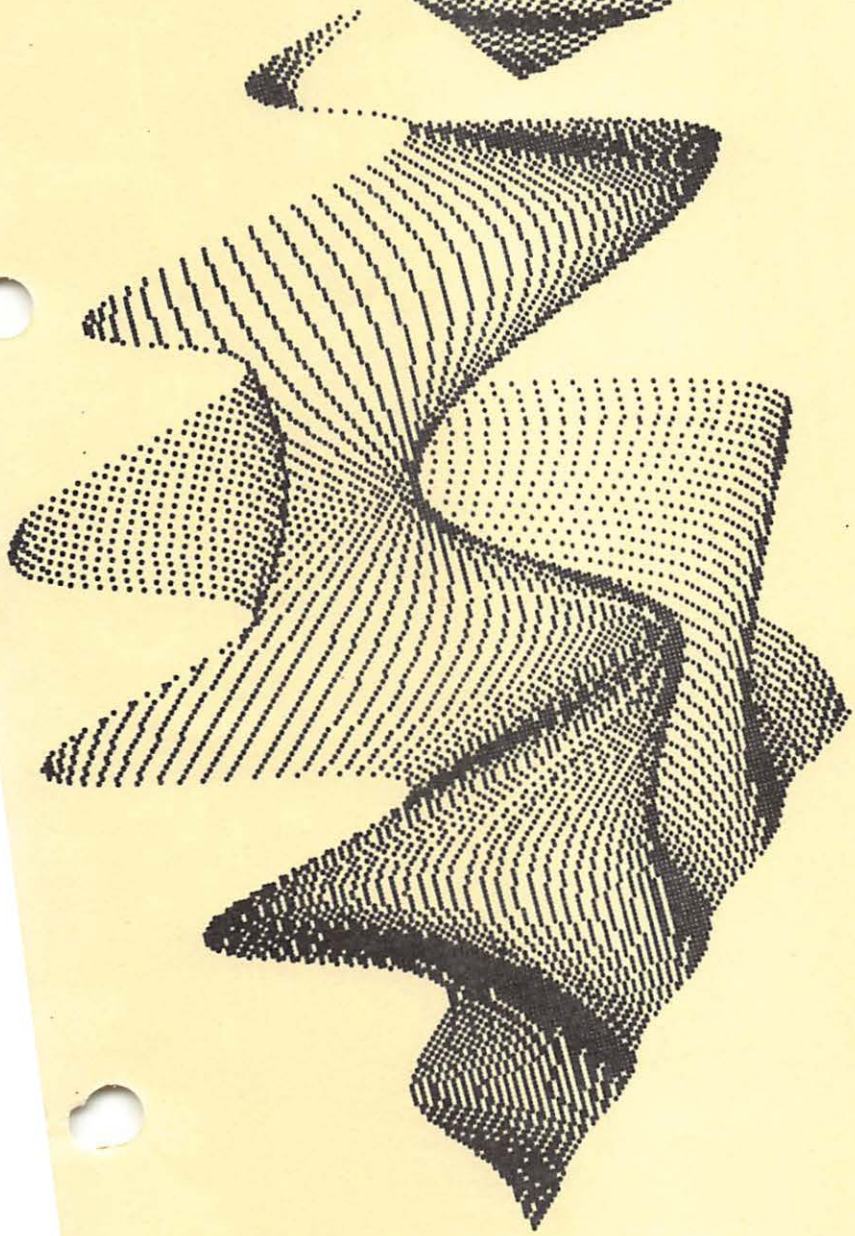
Shove in a "BSAVE...", etc., at line 345 and link to Apple's SLIDESHOW program

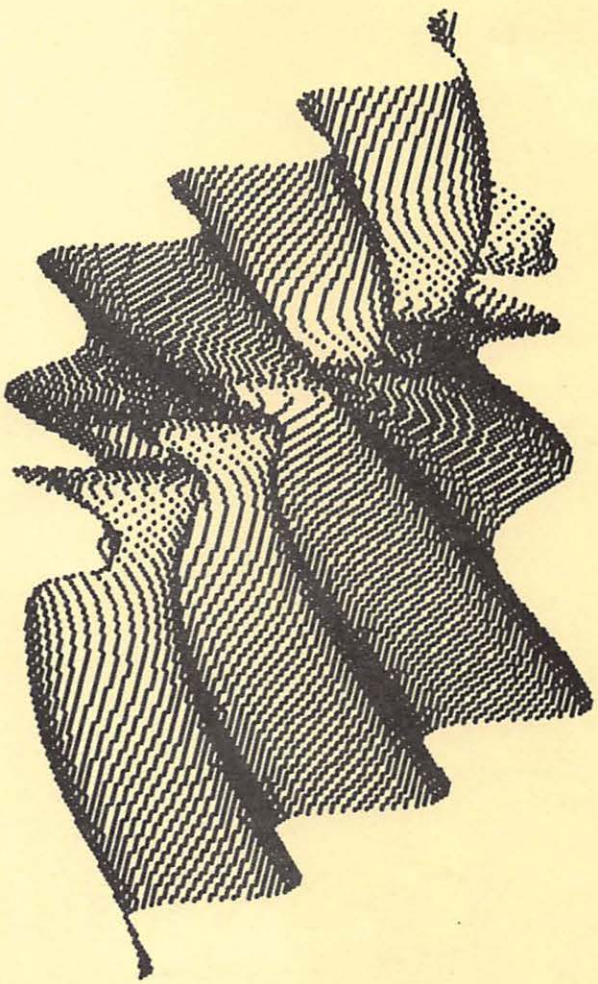
Experiment with the functions and the controlling parameters above

If you don't dig math, fear not- just run it and enjoy!

```
1  REM      ******************
2  REM      *                *
3  REM      *   PLOTPOURRI   *
4  REM      *   S COTTRELL   *
5  REM      *   SEPT  1979   *
6  REM      *                *
7  REM      ******************
8  REM
9  REM
10  TEXT
20  HOME : INPUT "WHICH PLOT FIRS
    T (1-10)?";PN
25  ON PN GOTO 30,40,50,60,70,80,
    90,100,110,120
30 X1 =  - 4:X2 = 4:Y1 =  - 4:Y2 =
    4:ZS = 45:RZ = 1:PC = 3:BC =
    0
35  DEF  FN Z(X) = ( COS (X * X +
    Y * Y) + 1) *  EXP ( - (X *
    X + Y * Y) / 6)
39  GOSUB 130
40 X1 =  - 4:X2 = 4:Y1 =  - 4:Y2 =
    4:ZS = 12.5:RZ = 1:PC = 6:BC
    = 0
45  DEF  FN Z(X) = 1 +  COS (X *
    Y)
49  GOSUB 130
50 X1 =  - 6:X2 = 2:Y1 =  - 5:Y2 =
    1:ZS = 25:RZ = 2:PC = 0:BC =
    3
55  DEF  FN Z(X) = ( -  COS (X *
    X + Y * Y) - 1) *  EXP ( - (
    X * X + Y * Y) / 16) + 1
59  GOSUB 130
60 X1 =  - 200:X2 = 200:Y1 =  - 1
    00:Y2 = 200:ZS = 12.5:RZ = 2
    :PC = 0:BC = 7
65  DEF  FN Z(X) = 1 +  COS ((X +
    Y) / ( LOG (X * X + Y ^ 4)))
69  GOSUB 130
70 X1 =  - 20:X2 = 20:Y1 =  - 20:
    Y2 = 20:ZS = 40:RZ = 2:PC =
    0:BC = 3
75  DEF  FN Z(X) = ( COS (X * Y /
    SQR (X * X + Y * Y)) - 1) *
    EXP ( - (X * X + Y * Y) / 1
    20)
79  GOSUB 130
80 X1 =  - 5:X2 = 5:Y1 =  - 5:Y2 =
    5:ZS = 12.5:RZ = 2:PC = 0:BC
    = 7
85  DEF  FN Z(X) = 1 +  COS ((X +
    Y) / ( LOG ( ABS (X * Y + .5
    ))))
89  GOSUB 130
90 X1 =  - 7:X2 = 7:Y1 =  - 17.4:
    Y2 = 17.4:ZS = 50:RZ = 2:PC =
    0:BC = 3
95  DEF  FN Z(X) = Y * X * X / (Y
    * Y + X * X * X * X) + .4
99  GOSUB 130
100 X1 =  - 3:X2 = 3:Y1 =  - 3:Y2
    = 3:ZS = 10:RZ = 1:PC = 2:B
    C = 0
105  DEF  FN Z(X) = Y * X * (X *
    X - Y * Y) / (X * X + Y * Y)
    + 2
109  GOSUB 130
110 X1 =  - 4:X2 = 6:Y1 =  - 3:Y2
    = 5.7:ZS = 32:RZ = 2:PC = 3
    :BC = 0
115  DEF  FN Z(X) = 3 *  EXP ( -
    ( SQR (X * X + Y * Y))) + .1
    *  COS (X * Y)
119  GOSUB 130
120 X1 =  - 2:X2 = 3:Y1 =  - 3:Y2
    = 5.7:ZS = 80:RZ = 1:PC = 6
    :BC = 0
125  DEF  FN Z(X) =  SIN (X * Y) *
    EXP ( - (X * X + Y * Y) / 9
    ) + .2
128  GOSUB 130
129  GOTO 30
130 NY = RZ * 87:NX = RZ * 50
140  HGR2
150 DX = (X2 - X1) / NX:DY = (Y2 -
    Y1) / NY
160 H0 = 0
170 K1 = 58:K2 = Y2 - Y1:K3 = 90
172 K4 = 174:K5 = 191
200  FOR X = X1 TO X2 STEP DX
220 H0 = (X - X1) / (X2 - X1) * 1
    00:NH =  - 1
240  FOR Y = Y1 TO Y2 STEP DY
250 ZZ =  FN Z(X)
260 U =  - ZZ * ZS - (Y - Y1) / K
    2 * K1 + K3 + H0
270 NH = NH + 1
280 H = H0 + NH * K4 / NY
290  IF U < 0 OR U > K5 GOTO 330
300  HCOLOR= PC: HPLOT H,U: HCOLOR=
    BC: HPLOT H,U + 1 TO H,K5
330  NEXT : NEXT
340  FOR J = 0 TO 3: PRINT  CHR$
    (7): NEXT
345  GET ST$: REM  HIT A KEY TO C
    ONTINUE
350  RETURN
```







COMPUTERS PLUS, INC.

6120 Franconia Road, Alexandria, Virginia 22310 703-971-1996

COMPUTERS PLUS, INC. carries a broad line of Micro Computers and peripherals as well as one of the largest book selections in the area.

Hours: 10:00 am – 9:00 pm Mon-Fri
10:00 am – 6:00 pm Saturday

Soon to be offering classes and seminars on all aspects of microcomputing.

Authorized dealers for: APPLE
Cromemco
Dynabyte
Micropolis
Northstar
soon to add: Thinker Toys

Authorized service: APPLE
Micropolis
Dynabyte

---

apple computer
Sales and Service

"The *Plus* Makes the Difference"

Washington Apple Pi
Membership Application

NOTE: Club policy prohibits revealing members' names and addresses. Additionally, the information requested below is for planning purposes only and will not be released to anyone, including other members.

NAME_____

ADDRESS_____

CITY, STATE, ZIP_____

TELEPHONE NUMBERS: HOME ( )_____ WORK ( )_____

PLEASE LIST HARDWARE YOU OWN:_____

_____

_____

OCCUPATION_____

I WOULD LIKE TO WRITE ARTICLES FOR THE NEWSLETTER (Y/N)_____

I WOULD LIKE TO ASSIST ON A COMMITTEE (SPECIFY AREAS OF INTEREST IF YES) Y/N_____

_____

PLEASE ENCLOSE PAYMENT WITH THIS APPLICATION IN THE AMOUNT OF $6 FOR 6 MONTHS

MONTH JOINED_____PAID (Y/N)_____

MAKE CHECK OR MONEY ORDER PAYABLE TO: WASHINGTON APPLE PI

SEND TO: WASHINGTON APPLE PI
PO BOX 34511
WASHINGTON, DC 20034

Washington Apple Pi
P.O. Box 34511
Washington, D.C. 20034

Third Class