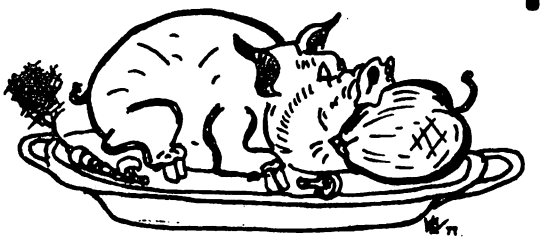
H.A.A.U.G.



HOUSTON AREA APPLE USERS GROUP

THE APPLE BARREL

< SINGLE COPY PRICE $1.00 >

==========

VOLUME 3 NO.8 NOVEMBER, 1980

President, Bruce Barber
Editor, Ed Seeger

==========

<<< CONTENTS >>>

<<< CLUB NOTES >>>

Houston Area Apple Users Group
APPLE BARREL
4331 Nenana Drive
Houston, TX 77035

The HOUSTON AREA APPLE USERS GROUP is an Apple II user club, not affiliated with Apple, Inc., or with any retail computer store. HAAUG is a member of the International Apple Core and supports its purposes and publications. General membership meetings are held on the second Thursday of each month in the rear chapel of Memorial Lutheran Church, 5800 Westheimer, right by the Jungman Branch Library and west of Chimney Rock. They start at 6:30 p.m. An additional meeting for access to the club software library, problem-solving, and various lectures is held the last Saturday of each month at the University of Texas School of Public Health (in the Medical Center), 6905 Bertner off Holcomb, across from M.D. Anderson Hospital. We meet on the main floor, first room on the left. Parking is adjacent to the building. These Saturday meetings begin at 2:00 p.m.

---==*==---

OFFICERS / EXECUTIVE BOARD

President	Bruce Barber	469-5805
Vice President	(vacant)	
Treasurer	Ray Essig	497-7165
Secretary	James Odom	426-3970
Software Lib.	Dennis Cornwell	774-0671
Hardcopy Lib.	Larry Baumann	498-3433
Hardware Chair	(vacant)	
Business Uses	Rudge Allen	622-3979
Membership	Lee Gilbreth	342-2685
Newsletter Ed.	Ed Seeger	723-6919

---==*==---

MEMBERSHIP INFORMATION

Dues are $18.00 per 12-month period for regular memberships, $6.00 for students through high school and where no adult member of the family is an Apple user. Please make checks payable to "Houston Area Apple Users Group," and mail to Lee E. Gilbreth, Membership Chair, 3609 Glenmeadow, Rosenberg, TX 77471. This includes a subscription to APPLE BARREL, which is published nine times a year. Newsletter exchanges with similar clubs are invited.

---==*==---

APPLE BARREL REPRINT POLICY

---==*==---

SPECIAL INTEREST GROUPS

Members who share a common interest are encouraged to form Special Interest Groups to more fully explore their fields. Meetings may be arranged by common consent of the group and will ordinarily have one member who serves to coordinate or convene the meetings. If you would like to start a group around any given interest, please contact one of the club officers. If you would like to be in touch with others who share one of the following interests with you, please phone the coordinator.

Current groups are:

1) BUSINESS APPLICATIONS
Coordinated by Rudge Allen, 622-3979

2) PASCAL USERS
Directory being assembled
Pat McGee coordinating, 663-6806

This Special Interest Group is to meet and discuss aspects of Apple's Pascal language and to exchange programs.

3) MODEM USERS
Directory being assembled
Herb Crosby coordinating, 497-1061

4) HAM RADIO OPERATORS
Coordinated by Ed Seeger, WB5PTW 723-6919

5) NEW MEMBERS
Coordinated by Lee Gilbreth, 342-2685

6) EDUCATIONAL APPLICATIONS
Coordinated by Darrell Kachilla, 498-0186

7) BEGINNERS' PROGRAMMING
Coordinated by John C. Whiteman, 974-7287 (home)
This Special Interest Group is to meet and discuss Integer Basic and Applesoft.

8) FILE CABINET
Coordinated by Lee Gilbreth, 342-2685
Purpose is to understand, expand and enhance the File Cabinet program.

---==*==---

APPLE BULLETIN BOARD SYSTEM

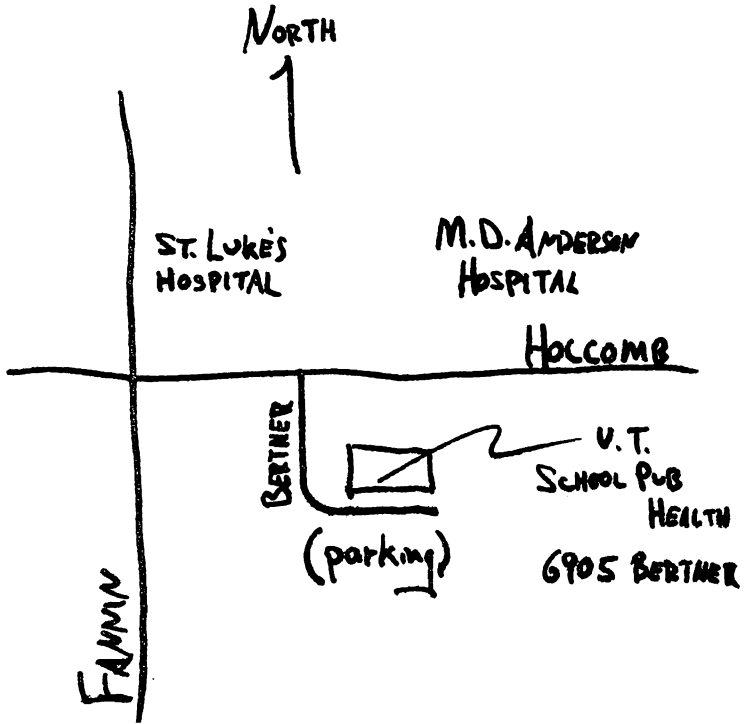
The Houston Area Apple Users Group supports an ABBS evenings and weekends, 6:00 pm through 8:30 am, and all weekend long. Feel free to sign-on and place your want-ad, meeting notice, request for help, Aggie joke, etc. Any ASCII terminal, Apple computer or not, with suitable modem or coupler, will give you ABBS capability. Note that our ABBS now has download capability! Call:

713/654-0759

SYSOP is Rudge Allen, 622-3979.

NEW SATURDAY MEETING PLACE

The informal meeting held on the last Saturday of each month has a new home. Beginning in December (Saturday the 27th.), we will meet at the University of Texas School of Public Health, 6905 Bertner off Holcomb. Look for us on the Main Floor, first room on the left, or follow the sweet sound of ctrl-G! This means we will no longer meet in the radio clubhouse. Thanks to the HAAUG Executive Board and to member Carl Hacker for arranging for these nice facilities. By the way, parking at the school is free and is right adjacent to the building.



C A N D I D A T E S

Elections for Houston Area Apple Users Group officers for 1981 will be held Thursday evening, December 11, during the regular club meeting. The nominating committee met and then submitted its report at the November meeting. Nominations from the floor were then received. Candidates are as follow:

President:	Bruce Barber, Lee Weitzenkorn
Vice President:	Mike Kramer, Rich Rennison, Bill Zahrt, Charlie Yust
Treasurer:	Ray Essig, Dick Gleason, Brian Whaley
Secretary:	Sam Block, Paul Maddock

All HAAUG members are urged to be present and to take part in the selection.

APPLE ORCHARD SUBSCRIPTIONS

P. O. BOX 2227 SEATTLE, WASHINGTON 98111, USA

The International Apple Core will make individual subscriptions to "The Apple Orchard" available commencing with Volume I, Number 2 to be published in September, 1980.

NAME _____

STREET _____

CITY _____ STATE _____ ZIP _____

COUNTRY _____

Annual Subscription Rate: $10.00 per year
First Class Postage: $5.00 per year additional (required for Canada, Mexico, APO, and FPO addresses)
Overseas and other foreign air mail postage (required): $10.00 per year additional

TOTAL REMITTANCE ENCLOSED: $(USA) _____

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

Apple Orchard Subscriptions
P.O. Box 2227
Seattle, Washington, USA 98111

7/7/80

NAME SWAP SUBROUTINE
BY
MIKE KRAMER

When using FILE CABINET I frequently enter a person's name as the first item in each record. Since reports look better with names printed first name first, the name swap subroutine in the listing below was developed.

The subroutine is actually fairly simple, involving a FOR...NEXT loop with a range equal to the number of characters in the entered name (NM$). The MID$ function in Line 10040 is used to search for a space (SP$). If no space is found, no action is taken except a RETURN. If a space is found, Line 10070 exchanges the first and last names, inserting a space between them, and returns to the main program.

```
LIST 0,10080
100 HOME : INPUT "ENTER NAME ";NM$
110 SP$ = CHR$ (32): REM  SPACE
120 GOSUB 10030
130 PRINT NM$
140 END
10000 REM
10010 REM NAME SWAP SUBROUTINE
10020 REM
10030 FOR X = 1 TO LEN (NM$)
10040 IF   MID$ (NM$,X,1) = SP$ GOTO 10070: REM  LOOK FOR SPACE
BETWEEN NAMES
10050 NEXT X
10060 RETURN
10070 NM$ = RIGHT$ (NM$, LEN (NM$) - X) + SP$ + LEFT$ (NM$,X -
1):REM FLIP-FLOP FIRST AND LAST NAMES
10080 RETURN

]RUN

ENTER NAME: MIKE KRAMER

KRAMER MIKE
```

PRINTER ACTIVATE/DEACTIVATE SUBROUTINES

BY
MIKE KRAMER

One of the most frustrating problems I've encountered with my Apple was trying to figure out how to tab past column 40 using the SSM AIO Interface Card RS232 port without evaporating my program. Some time back good old Dr.Apple helped by pointing out that I had to disable the screen before doing the tabs. Unfortunately, old Doc failed to tell how to do this. After a call to SSM's technician, I came up with the following subroutines to activate and deactivate the printer. Assuming the card is in slot #1, The POKE 1401,128 in Line 10120 disables the screen, but only if a character has been printed following the PR#1 command. In this example, a TOP OF FORM character has been output. The POKE 33,33 in Line 10130 is needed to permit printing past column 40. The deactivate subroutine puts memory locations back to their normal values and switches output back to the screen.

```
10094  REM
10095  REM ACTIVATE PRINTER
10096  REM
10100  D$ = CHR$(4):REM CTRL D
10110  PRINT D$;"PR#1"
10120  PRINT CHR(10):REM LINE FEED
10130  POKE 1401,128:REM DISABLE SCREEN
10140  POKE 33,33:REM SET TEXT WINDOW
10150  RETURN
10154  REM
10155  REM DEACTIVATE PRINTER
10156  REM
10160  POKE 33,33
10170  POKE 1401,0
10180  PRINT D$;"PR#0"
10190  RETURN
```

Apple Computer Registers Its First Offer, Of 4.5 Million Shares at $14 to $17 Apiece

By MARILYN CHASE
Staff Reporter of THE WALL STREET JOURNAL

CUPERTINO, Calif.—Not since Eve has an apple posed such temptation. But this time it's Apple Computer Inc., which filed its long-awaited registration with the Securities and Exchange Commission for an initial public offering of 4.5 million common shares. The tentative offering price will be between $14 and $17 a share.

The hot manufacturer of personal computer systems for the home, business and educational markets, said the offering is planned for early December. Apple announced its intent to go public last August, and investor appetite for the new issue has been keen ever since.

The company will sell four million shares, and certain holders will sell 500,000 shares. Morgan Stanley & Co. and Hambrecht & Quist will lead a group of underwriters. As previously reported, Apple said it plans to use proceeds mainly for working capital.

The registration statement filed yesterday charts the company's rapid rise since its founding in 1977 to become No. 2 in the growing field of personal computers, right behind Tandy Corp.

Net sales in fiscal 1979 roughly quintupled from fiscal 1978, the company's first full year of operation. Revenue rose to $47.9 million from $7.9 million, and earnings soared to $5.1 million, or 12 cents a share, from $793,497, or three cents a share.

In fiscal 1980, ended Sept. 26, profit more than doubled to $11.7 million, or $4 cents a share. Revenue rose to $117.9 million.

Published estimates of the company's performance call for revenue of $150 million in calendar 1980, and $300 million in 1981. However, the company will neither confirm or deny such estimates. "We're constrained from saying anything which could be construed as promotional," by the SEC, explained Frederick Hoar, vice president, corporate communications.

The company has 48.4 million shares outstanding, of 160 million authorized.

Apple's SEC filing, which offers the first inside glimpse of one of the most sought-after new stocks of the year, also depicts rising expenses for marketing, expansion and promotion.

Marketing expenses rose to 10% of net sales in fiscal 1980, reflecting additions to Apple's sales force. Apple said that such outlays are expected to continue.

Advertising more than doubled to $4.5 million in fiscal 1980, the company disclosed.

Apple also disclosed some potential soft spots. The filing acknowledged the company lacks the market penetration of Tandy Corp.'s Radio Shack and the broad distribution of Commodore International Ltd., which also sells personal computers. Apple said it "may also be at a competitive disadvantage" because it purchases integrated circuits and other parts used in computers, as well as a substantial portion of peripheral equipment, rather than making its own.

Positioning themselves for a share of the personal computer market are several larger companies including Hewlett-Packard Co., International Business Machines Corp., Texas Instruments Inc. and various Japanese manufacturers. And Apple said it is bracing itself for "intense competition" from such heavyweights.

Its newest weapon against such competition, the Apple III computer, has met with production delays, and first retail deliveries aren't expected until late this month, Apple said. Designed primarily for more sophisticated business applications than either the Apple I or the Apple II, the Apple III will range in price from $4,300 to $7,800, depending on main memory size and peripheral equipment.

Apple also disclosed that it is the target of a June 1980 lawsuit filed by a former distributor, High Technology Inc., which alleged violation of federal antitrust laws and breach of contract among other charges. The suit alleges several overlapping claims, the largest for $11,750,000 damages. The company has denied the allegations and filed a counterclaim.

Nautilus Fund Finds Cause for Uneasiness In 'Rush of Success'

* * *

Outlook for High-Technology Shares Troubles a Holder Of Apple Computer Stock

By a WALL STREET JOURNAL *Staff Reporter*

BOSTON—Nautilus Fund, the high-flying closed-end investment company, is getting worried about the outlook for its portfolio of high-technology stocks.

The company, whose shares have soared because it owns 180,000 restricted shares of Apple Computer Inc., is "delighted with the rush of success," Albert L. Toney, president, said.

"But the suddenness of its arrival during a cyclical correction in the economy, and the lofty heights to which stock prices of emerging companies have climbed, cause unease," Mr. Toney told the fund's annual meeting.

Because of this, the fund has "cut back on a number of positions during the last month," he said, and has withdrawn 15% of its cash from the market and put it into reserves. At the same time, Nautilus has "become even more selective in making purchases," he added.

Nautilus's holdings in Apple Computer are restricted shares, which means they can't be traded. Nonetheless, mainly because Apple plans a public stock offering later this year, the stock price of Nautilus last week was 57.5% above its net asset value.

Mr. Toney had said Nautilus plans to register its Apple shares with the Securities and Exchange Commission after the maker of personal computers issues stock to the public. But the Nautilus board, in an apparent reprimand, later issued a statement saying that it—rather than Mr. Toney—would decide the future of the fund's Apple shares.

Besides Apple, the major holdings of Nautilus include MCI Communications Corp., Paradyne Corp., Gerber Scientific Inc., UTL Corp., Evans & Sutherland Computer Corp., NBI Inc., Yellowknife Bear Mines Ltd., Anacomp Inc. and Southwest Airlines.

Overall "the outlook continues good for the dynamic companies we follow, but expectations of company management are becoming more guarded," Mr. Toney said. "The recent surge in business activity

Reprinted from the
Wall Street Journal
October 20 & November 7, 1980

WANTED: Person for full or part time position programming Apple II Plus. Must be proficient in Applesoft and Integer. Experience in Education and/or Aviation Fields helpful but not necessary.

This position would involve writing educational courseware for a helicopter training company, using the latest in Technology including video tape and disk interfaces.

Salary negotiable.

Call Joel Harris at 353-6540(Houston) or 539-1893 (Conroe).

PATS PASCAL CORNER

The first four disks of the UCSD Pascal Users Group Library have finally arrived. Programs of interest include a fancy text formatter, two Pascal prettyprinters, a simple text printer, Othello, blackjack, a sorter for ASCII files, chase, and a home finance program. These disks will be available from me at the saturday swap session. The next four disks will include an index for Jensen & Wirth, Ken Bowles' new database seed, Wumpus (with 6 different caves), a file comparison program, a flexible data base/mailing list program, a disk patcher, a program to change identifiers in a source program, subroutines to convert from/to hex,decimal and octal, and several other software tools. I'll let you know when these come it, probably around the end of September. I've paid for these disks ($90 so far) and am accepting contributions to spread the burden around.

Pat McGee

VOLUME ONE, UCSD PASCAL USERS' GROUP -- CATALOG

NOTE WELL: Let it be said here for all the files on this disk that UCSD Pascal is a trademark of the Regents of the University of California. All software on this disk may be given away but NOT sold without prior arrangement with SofTec and/or Datamed Research.

CATALOG1.TEXT.....what you're reading now.
COMBINE.TEXT.....a simple little thing to combine 2 or more text files.
CRT.I.O.TEXT.....very powerful, crash-proof data entry UNIT for CRT menus.
FORMAT.DOC.TEXT.....documentation (from Pascal News) for FORMAT.
FORMAT.TEXT.....large, wonderful Pascal program prettyprinter.
FORMAT1.TEXT.....part of FORMAT.TEXT (subfile).
FORMAT2.TEXT.....part of FORMAT.TEXT (subfile).
INITVAR.TEXT.....part of PRETTY.TEXT (subfile).
L.TEXT.....a short but effective text printer with several options.
PRETTY.TEXT.....the second Pascal prettyprinter, from the Pascal News.
PRETTY.DOC.TEXT.....documentation for both FORMAT and PRETTY.
SIMP.TEXT.....cute program to produce random text; sounds "professional."
TYPESET.TEXT.....takes text from editors & right-justifies it.
UNITS.DOC.TEXT.....re UNITS, SEGMENTS, & EXTERNAL routines.
VOLUME1.TEXT.....how this disk is organized (more detail).

Have fun! Let me know if you spot bugs or errors in any software or documentation on this disk, or if you can clear up further mysteries of UCSD Pascal.

Jim Gagne, DATAMED RESEARCH

APPLE VOLUME 2 CATALOG, UCSD PASCAL USERS' LIBRARY

PASCAL TRANSFER PROGRAM and other goodies.*

ACOUSTIC.TEXT.....Use an acoustic modem with the Pascal Transfer Program (PTP).
DCHAYES.IO.TEXT....Use a D.C. Hayes modem w/ the Pascal Transfer Program (PTP).
DELETE.LF.TEXT.....After transfering a textfile to UCSD, dumme ASCII linefeeds.
HEXOUT.TEXT.....Pascal routine to print out integers in hexadecimal.
KBSTAT.TEXT.....Yet another keyboard status routine, this time for PTP.TEXT.
LINECOUNTR.TEXT....Count the lines of a textfile.
NEW.GOTOXY.TEXT....Good idea: let GOTOXY handle your CRT screen, too. Sample.
PERUSE.PG.TEXT.....Look over a textfile on your CRT one page at a time.
POLICY.DOC.TEXT....How the Users' Group Library runs.
PRIME1.TEXT.....Pascal routine to find prime numbers.
PRIME2.TEXT.....Another prime-number generator.
PTP.DOC.TEXT.....Documentation for the Pascal Transfer Program.
PTP.TEXT.....The Pascal Transfer Program. Requires L2 editor to edit.
SMARTREMOT.TEXT....Set up your machine as a smart remote terminal.
UPDATE.DOC.TEXT....Latest news on the UCSD Pascal Users' Group Library.
VOLUME.2.TEXT.....Notes on all the programs in Volume 2.
WRITER.DOC.TEXT....Documentation for WRITER.
WRITER.TEXT.....A quick but nifty text or source file printer.

* Note: UCSD Pascal is a trademark of the Regents of the University of California. Please read the file POLICY.DOC.TEXT regarding the software on this disk. All files are further documented in VOLUME.2.TEXT.

APPLE USERS NOTE: VOLUME.2.TEXT DESCRIBES ALL THE FILES ON THE ORIGINAL TWO 8-INCH DISKS (VOLUME 2A AND VOLUME 2B), WHICH I HAVE LEFT FOR YOUR PERUSAL. ALSO, FOR NOW THE ASSEMBLY-LANGUAGE ROUTINES FOR THE D.C. HAYES MODEM ARE IN 8080 ASSEMBLY LANGUAGE AND ARE FOR THE OLD S-100 D.C. HAYES. YOU'LL HAVE TO MODIFY THEM ON YOUR OWN. SHOULD BE EASY.

** FLASH **
THE UCSD SYSTEM USERS SOCIETY, FORMED 21 JUNE 1980, SELECTED THE PASCAL TRANSFER PROGRAM AS ITS OFFICIAL INTERMEMBER MODEM DRIVER. IT'S SLOW, BUT IT IMPLEMENTS WHAT IS KNOWN OF THE PCNET PROTOCOL, AND LETS YOU TRANSFER FILES IN TWO DIRECTIONS AT ONCE, WHILE ALLOWING CONVERSATION ON YOUR TERMINALS, <SIMULTANEOUSLY>!! MINOR MODIFICATIONS ARE PENDING (PRIMARILY IMPROVED USER INTERFACE) IN A FEW MONTHS.

VOLUME 3 CATALOG, UCSD PASCAL USERS' GROUP LIBRARY

Prose, games, and some ideas.*

BLACKJACK.TEXT.....Now you can play it in Pascal. Appropriate for 1980: allows negative money.

CHASE.TEXT.....A good implementation of an old favorite. Get away from the robots, but don't get zapped by the electric fence!

DEBTS.TEXT.....Home finance program, keeps track of your bills. Nicely menu driven, easy to use.

OTHELLO.TEXT.....VERY nice implementation of OTHELLO, the best I've seen.
OTHELL1.TEXT
OTHELL2.TEXT
OTHELLINIT.TEXT....Subfiles of OTHELLO.

POLICY.DOC.TEXT....How the Users' Group Library works.

PROSE.DOC1.TEXT
PROSE.DOC2.TEXT....A subset of the documentation of Prose, copied from the Pascal News No. 15. What you really need to know to use it.

PROSE.TEXT.....A copy of the fancy text-formatting program from the Pascal News, No. 15, adapted for UCSD Pascal by its author, J. P. Strait, of the University of Minnesota. Requires most of 64K of memory to compile.
PROSE.0.TEXT
PROSE.A.TEXT
PROSE.B.TEXT
PROSE.C.TEXT
PROSE.D.TEXT
PROSE.E.TEXT
PROSE.F.TEXT.....Subfiles of Prose.

PROSE.I.5.CODE.....Object version for those without sufficient memory to compile; will run under UCSD versions I.4 and I.5.

REQUESTS.TEXT.....Some ideas for some very needed programs and routines.

SNOOPY.TEXT.....Snoopy calendar, featuring the W.W. I flying ace.

STORE.DATA.....Sample data file for DEBTS.TEXT.

UNIVERSAL.TEXT.....Suggestion for a UNIT that will let us use each other's programs without having to edit in hardware-specific routines.

* NOTE: UCSD Pascal is a trademark of the Regents of the University of California. Please read the file POLICY.DOC.TEXT regarding the software on this disk. All programs should be self-documenting, though you'll have to fix hardware-specific procedures in the same programs (see UNIVERSAL.TEXT for a discussion of this subject); as a rule, any code your system does not support (e.g., KeyPress or a system clock) can just be deleted.

CCA DATA MANAGEMENT USERS GROUP

I had an interesting talk on the phone not long ago with Ben Herman, author of the CCA Data Management System, distributed by Personal Software. I know that a number of our HAAUG members have bought this package and so should be happy to know that a users group has formed. Ben reports that the group has just under 500 members now and has put out two newsletters.

Because the programs are written in Applesoft, and are therefore user-accessible and modifiable, there is a definite need for information exchange about customization, if nothing else. But beyond this, the system has spawned several "satellite systems," programs and documentation which are to be used in conjunction with it. VisiCalc is obviously one of these satellites, since the CCA menu hooks directly to VisiCalc interface routines. Also becoming available are a home accounting system, and also a memory-mapped screen editing system written in assembly language and replacing the file maintenance module currently residing on the system disk. There is to be a file fixer which will retail for $9.95, and a disk catalog system. Look for a multi-key scan facility which implements compound logic for file searches. It will handle up to twenty-four levels of selection, although its and/or logic is reported to be limited. And there will be more spin-offs as well. Apple users who have not yet probed the power of the Indexed Sequential Access Method of file searching (ISAM) have in the CCA system a way to learn about this approach to filing and searching. If you, like me, view your Apple as much as a teacher as a tool, CCA has much in store for you, and this new users group should make the way a good bit easier-going.

Colin Jameson, of Jameson Electronics in Los Angeles, is editing the newsletter, which appears more or less quarterly. Membership dues are $9.00 per year, and confer a newsletter subscription, use of both an east coast and west coast hotline number, and one free CCA-related ad. Sounds good! Colin can be reached at 1-213-540-5208. While back issues of the newsletter are available, they are sent only with a 10-month membership, rather than 12, since this keeps records a lot more manageable. Let Colin handle that if you do decide to join. You will be billed after receipt of membership materials.

—Ed Seeger

SPECIAL PRESENTATION FOR DEC 11 MEETING

The December 11 meeting will feature a demonstration by Watanabe America of their Model WX4671 MIPLOT Intelligent Plotter. You may have seen the impressive demo of the MIPLOT on a PET (sorry) at Watanabe's booth at the ISA Show. This will be a good opportunity to see a peripheral which is not yet in common use on personal computers in the Houston area.

In conjunction with Watanabe's presentation, B.P.I. of Austin, developers of an extensive line of business-oriented software for the Apple II, will demonstrate their inventory, general ledger, accounts receivable, and accounts payable packages.

HAAUG BUSINESS GROUP MEETINGS

Time: 7:00 p.m.

Place: EBASCO offices
3731 Briar Park at West Park
Houston

Thursday, December 4, 1980

Charting Commodity and Stock Prices with the Apple
Prices retrieved via Modem.

Thursday, January 15, 1981

Demonstrations and Reviews of Data Base Management Systems including Personal Software's CCA DMS, Data Factory File Cabinet, Modifiable Data Base, and others.

For Further Information Call:
Rudge Allen, 622-3939

<<< DOS 3.2 DISASSEMBLY >>>

We continue in this issue our sixth installment of Lee Meador's excellent series on the Disk Operating System, as originally published in the "Fort Worth Apple Users Group Newsletter." This installment is taken from vol. 1, no. 8, 15 May, 1980. Lee is thinking of preparing a technical booklet on Apple DOS, with these studies as the core. Comments, errors noted and suggestions can be directed to him at 1401 Hillcrest Drive, Arlington, TX 76010.

DOS Disassembly (Part 5)

by Lee Meador

Surprise! The title for this installment is wrong. This is really information about how the Apple Disk II controller works. I just used that title to make this a part of the same series. Oh, well.

We all know that the hardware that runs the Apple II Disk is non standard. That is why you can't go down and but a $300 disk drive and plug it into the Apple controller. Let's try to reconstruct the thinking that went into the decision to make instead of buy the controller. You have to remember that this is taking place several years ago. Disk controller chips were fairly expensive and 5" disk drives were fairly new on the scene. A single drive with controller ran around $1000 at the least and Apple saw that people would be a little reluctant to buy a thousand dollar disk drive to go on a thousand dollar computer. A lot of people would want two drives and that put the cost out of line for the people who were buying Apple II computers. We bought an Apple because it was a lot cheaper than the competition anyway. (Don't forget the TRS 80 was not yet available.) So, Apple made a deal with Shugart, the people who make disk drives, and bought a bunch of drives with no controller electronics inside. Then Apple can come up with a brand new method of building a disk controller that takes about one fifth the number of chips and, accordingly, costs less. And so the Apple II disk controller was born. Soon, the 5" drive became more popular and as the number being made increased the cost came down. Costs also came down on controller chips. By now the Apple Disk II drives are no longer significantly cheaper than the standard ones. At least there isn't much difference when we buy them. There really isn't any competition for Apple and they have no reason to bring their price down without it. (*Ed. I did hear about LOBO offering or soon offering a plug compatable drive without controller card.*)

If you look on page 145 in your DOS manual, you will see the schematic for the Apple Disk II interface card. This is the card you plug into slot 6 (or whatever). It has places where one or two cables can be plugged in. The cables are shown running off the top of the page. Page 146 is the schematic for the Apple Disk II analog board. This is the board that is inside the metal cabinet that houses the disk drive itself. If you remove the metal cover by taking out the four screws in the bottom of the drive you will see this card right on top. (Slide the top cover off to see it.) Both of these boards are made by Apple

Computer Co. The rest of the drive is made by Shugart and sold as a package to Apple. These parts are standard to all Shugart drives as far as I know.

Here are some interesting things you should note about the schematics. First, consider the Interface card. The references below are looking at page 145 sideways so the printing will be right side up. The chip at the lower left is an 8 bit addressable latch. The input marked D is the data. The Device Select line is the clock for the latch. The inputs marked ABC select one of the 8 outputs marked Q. R resets all lines to low. When the Device Select line goes low (ie. You address one of the 16 device control addresses -- $C0x0 to $C0xF where x is 8 plus the slot number) the value of D is transfered to the appropriate Q output. The other Q outputs are not changed. This allows the you to step the head in or out by changing Q0 through Q3, select drive 1 or 2 with Q4, turn the motor of the selected drive on or off with Q5 and do things that will be discussed later with Q6 and Q7. Notice that when the motor is turned off the various mini-processor chips (I'll define that later, its the three chips in the top center) are actually disconnected from the power supply. This is to keep the power requirements low. Notice that the IO Select line turns the power on for the P5 Rom in the same manner. The ROM is fast enough to be turned on at the power pin, catch the address and return the desired data before the Apple needs it. (By the way, memory on peripheral cards only needs to be half as fast as the memory on the main board because it is not used to refresh the screen.) Another interesting thing about the P5 ROM is that the Address lines are mixed up and so are the data lines. The choice of numbers for particular data and address lines on a memory chip is really arbitrary, as long as you use the same conventions going in and coming out. The manufacturer always picks an order but Apple choose to ignore it. They probably did that to make the traces on the board easier to lay out.

Now for the mini-processor, as I have named the three chips in the top-center of the drawing along with a few other gates and wires. The P6 ROM chip and a 4-bit latch form the heart of the mini- processor. They provide the information to make the shift register do what is necessary to interface to the 6502 data bus. The latch holds the output of some of the output lines from the ROM and feeds them back in to allow the ROM to step from state to state. I define a state by the data on the address lines of the ROM. The ROM is laid out so that it loops through a limited number of states if there are no changes in the inputs to the mini-processor. It might change the outputs in the process. The inputs are Q6, Q7, Write Protect and Read Data. The output is Write Data. The 8-bit value in the shift register can be both written and read. This mini-processor uses the PHI3 (I have no greek letters) clock so it runs at 2 MHz. It changes state twice as fast as the 6502 processor clock. Thus 64 cycles of the mini- processor are the same as 32 cycles of the 6502. This, incidentally, is the amount of time needed to write one byte to the disk.

The Analog Card is a mystery to me. I have been told that the large chip in the bottom right hand corner of the drawing on page 146 is hooked up in a standard configuration. It is supposedly right out of the Motorola book. I have neither confirmed that nor proved it wrong. You see, I get lost easily when talking about hardware.

Let's talk about the Q6 and Q7 lines for a moment. You have seen them referred to in the RWTS listings and in the disassembly from the last issue. They control what the mini-processor is doing.

Q7	Q6	
0	0	- Read into Shift Register from diskette
0	1	- Check Write Protect status
1	0	- Write Shift Register to diskette
1	1	- Load Shift Register from Data Bus

How to do a write: You load a value to write into the shift register by bringing Q6 high and then do STA $C08F,X to bring Q7 high. (You could have Q7 high and do STA $C08D,X if you want.) Then do LDA $C08C,X (or ORA as another option) to bring Q6 low and continue the write. After the value is loaded (the mini-processor waits 5 mini-cycles and), the write line goes high (when bit 7 is high and it should always be high since you can't read bytes back in if bit 7 is low.) After that something will happen every 8 mini-cycles. If there was a one bit as the high bit of the shift register, then the write line changes (low-high or high-low). If there was a zero bit as the high bit of the shift register, then the write line stays the same. (ie. What happens is nothing at all.) The shift register is shifted left once every 8 mini-cycles so that the change or no-change in the write line reflects the contents of the shift register. On the 3rd mini-cycle after the write line changes (or doesn't change) the shift register is shifted left and the next highest bit is available. Then 5 mini-cycles later the write line reflects the new value of QA (ie. the latest high byte in the shift register.) See the WRITE chart for a look at the states the mini-machine goes through when it is writing bytes. The contents of the P6 ROM that affect the Write states have not been changed from the old to the Pascal version of the P6 ROM. The Disk software is responsible for putting a new value in the shift register or turning off the write mode exactly 64 mini-cycles after the previous byte was put into the shift register. If not, extra zero bits are written on the diskette following the eight bits from the last byte. That is why each byte that is written must have the 7th bit set. A zero bit is the default—no change. That is also why the software to produce self-sync (also called auto-sync.) writes an $FF to the disk but waits 36 cycles (ie. 72 mini-cycles) before writing another byte. Notice what this will do in the Read algorithm below.

How to check Write protect: When the Q6 line is high and the Q7 line is low the mini-processor will change into state $14 within 2 mini cycles. State $14 shifts right (bringing the value of the write protect line from SR into the low bit of the shift register) and then goes to itself. It only takes 8 mini cycles before the whole shift register reflects the value of the Write Protect line. $FF means the slot on the diskette is covered and zero means it isn't. Within those 4 processor cycles there isn't even time to read the shift register before it's ready. So, LDA $C08E,X will get $FF or zero from the shift register and a BMI will branch if the disk is write protected. X should hold the slot number in the form $x0. The mini-processor stays in state $14 until either Q6 or Q7 changes. There were no changes made in this part of the ROM when they made the Pascal P6 ROM, either.

How to do a Read: To me, reading is the most interesting part of this whole thing. It's because when you are reading you must reproduce in the shift register whatever data was written. That isn't easy because you may be reading on a different drive from the one that wrote the diskette. Not all drives are created equal, some run slower, some faster. Even if it is the same drive there may be slight voltage level differences that cause changes in the motor speed. It might be warmed up now and run a little faster or slower. The room temperature or humidity might be different causing the diskette itself to expand or contract. That would cause the read head to be a little off from the center of the recorded signal. If the diskette is a little off center on the spindle, the head will weave back and forth over the center of the recorded signal causing it to vary in strength like a warped record does. The signals would be weakened and distorted by age or lack of care. Magnetic fields of all sorts and even cosmic

rays all combine to make the data on the diskette a little distorted from the way it was written.

So, how does it manage to work so well. We start with something in the shift register. It's whatever was the last byte on the diskette. The read head gives us pulses corresponding to the changes in the write line described above. A pulse means there is a one bit and no pulse means there is a zero bit. The trailing edge of the pulse is all that matters. Now, as we start the first trailing edge will clear the shift register (after a few cycles) and put a one bit in for that trailing edge. If in the mean time we got anothere trailing edge we shift in another one bit. If we didn't we immediately shift in a zero bit. Then we get in the main loop of read states. It takes 3 mini-cycles (thats 1500 nanoseconds) to process a trailing edge and be ready for the next one. They should happen exactly 8 mini-cycles apart under ideal conditions (ie. drive is same speed, no distortion, etc.) and cause a one bit to be shifted in at state $F0. On the other hand, if the mini-processor gets to state $B0 without getting a trailing edge then a zero bit is shifted in. That means ten cycles without a trailing edge are what the machine uses to say it found a zero bit. When the last bit is shifted into the shift register (under normal conditions) then the high bit will become a one and a LDA $C08C,X can read a negative byte from the disk mechanism. That means the data is valid. Since the high bit of the shift register is a one, the mini-machine now goes into state $12 and waits for the next trailing edge. (I figure it usually stays there about 5 mini-cycles) When it finds the edge we start the whole process over again. There are about 6 processor cycles (12 mini-cycles) between the completion of the byte formation and the clearing of the shift register for the next byte. If the 6502 reads again too soon it will get the same byte again. If it waits too long (that is 38 processor cycles or 76 mini-cycles under ideal conditions) it will miss a byte. Notice that reading is synchronized by the formation of negative values in the shift register and the leading high bit (or the trailing edge of the corresponding pulse from the read head) of each byte. It does not depend on timing loops.

This is a good place to explain auto-sync bytes. They are written as 8 one-bits in a row followed by a zero bit by the 6502 waiting 72 mini-cycles between shift register loads. When reading begins the mini-machine does not know where it is within the byte. It could begin at bit 4 since the bits are stored as one long line of pulse-no-pulses. Auto-sync pulls the mini-machine in syncronization with the way the bytes were read because the mini-machine will read 8 bits and ignore the 9th only if the 9th is a zero bit. Otherwise the 9th bit will be taken as the start of a new byte. Eight or more bytes of auto-sync are written between sectors and between the sector header and the data portion. That is enough to bring the mini machine in sync with the real bytes.

There are changes in the read section of the Pascal P6 ROM that allow for more leeway in reading. It looks to me like the Pascal P6 ROM is really just changed to correct a defeciency in the original P6 ROM. (See the comments below).

Why does the software translate all the data into 5 bit nibbles and then into 8 bit values that don't have two zero bits in a row? (Last months installment of this series shows the software that does the conversion.) Suppose we tried to read two zero bits in a row. If they are written every 8 mini-cycles then the 4 bits "1001" would have the two trailing edges (for the one bits) 24 mini-cycles apart. It takes 3 mini-cycles to recover from the first one bit so there are only 21 mini-cycles in which to detect two zero bits. But, it takes 20 mini-cycles without a pulse to signal two zero bits in a row and that leave one mini-cycles as a margin for error. This would, in my opinion, work if the drives ran at the same speed. If the diskette was written on a drive that was more than 5% faster than the one it is being read on, then both zero bits would not be detected. The Pascal system writes double zero bits and reads them correctly and you it is possible to run Pascal with the old P6 ROMs. I know people who are doing it on a second controller with a third drive on it. It just means their drives are close enough to being the same that the double zero bits can be read.

Just how is the Pascal P6 ROM different? The Pascal mini-processor ROM has 5 bytes changed from the old P6 ROM. The listings below were produced by swapping the P6 and P5 ROMs on an extra controller card and moving the contents down to RAM memory. I've saved these as B type files on a disk. (By the way, you need to enter them to run the Basic Modeling Program.) What they do is change the action done in states $03, $13, $92, $82 and most importantly in $B0. $82 is a dummy state that is never gone to. The change in $B0 seems to have precipitated the other changes. The change in $B0 makes the mini-processor skip the first two states in the usual loop after finding a zero bit. That means the first zero bit is signaled by ten empty mini-cycles and the following zero bits only take 8 empty mini-cycles. With this change we can have two zero bits in a row and still read what was written on drives of slightly different speed. If we were to write more zero bits it would increase the chances of getting lost. Besides there is nothing to gain. (More about that later.) The change in $B0 means you have to change $03 and $13 to act just like $02 and $12 did before. Wait in state $12 or $13 after the shift register is full until another trailing edge comes down the line. Then go to state $02 or $03. The only difference is that you go to $12 if the last bit (low order) is a one bit and go to $13 if the last bit is a zero bit. Both $02 and $03 go to state $92 and $92 goes to $93 or $83 depending on the absence or presence of a trailing edge. These are the only changes made to the old P6 ROM when it became the Pascal P6 ROM. They allow you to read the double zero bits that you always could write.

Pascal also encodes the data differently from the way the Apple DOS 3.2 encodes it. Pascal encodes a page of 256 ($100) bytes as 343 ($157) groups of 6 bits. When being written the 6 bit nibbles are mapped into 64 of the 256 possible 8-bit patterns. These are the conditions on the 8-bit patterns used by Pascal to store onto the disk surface: Bit 7 must be on and no three zero bits may be together. DOS 3.2 encodes a page of 256 ($100) bytes as 410 ($19A) groups of 5 bits. These 5 bit nibbles are mapped into 32 of the 256 possible 8-bit patterns. (The conditions on the the 8-bit patterns used by DOS 3.2, which are repeated from the last installment, are: Bit 7 must be on and no two zero bits may be together.) Because Pascal uses 343 and DOS 3.2 uses 410 bytes to encode the page, Pascal has room on a track for 16 sectors instead of the normal 13 for DOS 3.2. (DOS 3.0 and, as I understand, the CP/A system by Shepardson Microsystems—who, incidentally, are rumored to be the people who wrote the original Apple DOS—use only 10 sectors per track. That gives 87.5 Kbytes per diskette as compared to 113.75K for DOS 3.2 and 140K for Pascal.) The bits aren't actually any closer or farther apart but the data is encoded more differently. If you think about it the only way to make it more efficient than Pascal (besides changing the whole hardware setup) is to encode bytes as 7 bit nibbles. That would use 128 of the 256 8-bit patterns. $D5 cannot map into a nibble (it's used to mark the front of a sector or sector head) and the high bit has to be set for the hardware to read correctly. So we need 129 8-bit values with the high bit set. That isn't possible (there are only 128 of them) so there is no need to try to write more than two zero bits in a row as it couldn't help us scrunch the data up further anyway.

The program given models the action of the mini-processor on the disk interface card. The screen

display shows the values of most of the lines shown in the schematic on page 145. You can change the inputs to the mini-processor—Q6, Q7, Read and Write Protect—by typing 6, 7, R or W, respectively, followed by RETURN. When you just hit return, one clock cycle is simulated and all the lines change appropriately. A trace of the last few states is shown at the right side of the screen. Most of the values are labeled according to the schematic. I have found this most useful when looking at the listings of the READ, WRITE and RDADR routines and the RWTS mainline and FORMAT program. It would help with the Apple COPY program, too. Just look at a table of execution times for the various instructions in your listing as you follow along in the disassembly. . Hit RETURN twice for each 6502 processor cycle since the mini-processor runs off the PH13 clock. Use the comments on the previous installments to decide when to change Q6 and Q7. You are on your own when deciding when to change the Read data line. I was unable to figure out just exactly how the changes in the Q6 and Q7 lines fit into the group of say 4 or 5 processor cycles taken to execute a LDA or STA. That has to fit in with when the shift register puts data on or accepts it from the 6502 data bus. I don't know enough about how the 6502 works. If someone could work this out and send timing charts I would be glad to disseminate the information.

I guess after all this I need a disclaimer. I have not been in contact with Apple Computer Co. about this. I understand they don't talk about it since there is a patent pending on the design. Everything in this article is either my idea or something told me by someone who knows something about hardware, software or the Apple rumor mill. I claim responsibility for the misteaks. Special thanks go to Tom Bonfield, Stan Brooks, Arlie Dealey and Kris whose last name I do not know. I also owe a lot to the articles listed on page A10 of the December 1978 issue of Computer Design. (In their terminology the Apple Disk II data is encoded using a modified form of Group Code Recording (GCR). GCR is usually a double density method but the Apple Disk II uses 4 microsecond bit cells instead of the normal 2 microseconds so it gets normal capacity by using double density methods at half speed.)

FWAUG Newsletter

Pascal P6 ROM Listing

courtesy of Arlie Dealey

```
*2000.20FF

2000- B8 B8 B8 C8 0A 0A 0A 0A
2008- B8 C9 B8 C9 B8 C8 B8 C8
2010- B8 C8 08 C8 0A 0A 0A 0A
2018- B8 C9 B8 C9 B8 C8 B8 C8
2020- B8 30 B8 B8 0A 0A 0A 0A
2028- 98 D9 98 D9 98 D8 98 D8
2030- 98 DD 98 D8 0A 0A 0A 0A
2038- 98 D9 98 D9 98 D8 98 D8
2040- B8 B8 B8 B8 0A 0A 0A 0A
2048- A8 E8 A8 E8 A8 E8 A8 E8
2050- A8 E8 A8 E8 0A 0A 0A 0A
2058- A8 E8 A8 E8 A8 E8 A8 E8
2060- B9 FD B8 F8 0A 0A 0A 0A
2068- B8 F8 B8 F8 B8 F8 B8 F8
2070- B9 FD 50 F8 0A 0A 0A 0A
2078- B8 F8 B8 F8 B8 F8 B8 F8
2080- B8 B8 48 B8 0A 0A 0A 0A
2088- 48 28 48 28 48 28 48 28
2090- 48 28 48 28 0A 0A 0A 0A
2098- 48 28 48 28 48 28 48 28
20A0- B8 B9 B8 B8 0A 0A 0A 0A
20A8- 58 38 58 38 58 38 58 38
20B0- 09 C9 58 38 0A 0A 0A 0A
20B8- 58 38 58 38 58 38 58 38
20C0- B8 B8 B8 B8 0A 0A 0A 0A
20C8- 68 08 68 18 68 08 68 18
20D0- 68 18 68 18 0A 0A 0A 0A
20D8- 68 08 68 18 68 08 68 18
20E0- 8D BD 78 70 0A 0A 0A 0A
20E8- 78 18 78 08 78 18 78 08
20F0- 0D 2D 78 70 0A 0A 0A 0A
20F8- 78 18 78 08 78 18 78 08
```

Old Style P6 ROM Listing

courtesy of Arlie Dealey

```
*2000.20FF

2000- B8 B8 B8 B8 0A 0A 0A 0A
2008- B8 C9 B8 C9 B8 C8 B8 C8
2010- B8 C8 08 48 0A 0A 0A 0A
2018- B8 C9 B8 C9 B8 C8 B8 C8
2020- B8 30 B8 B8 0A 0A 0A 0A
2028- 98 D9 98 D9 98 D8 98 D8
2030- 98 DD 98 D8 0A 0A 0A 0A
2038- 98 D9 98 D9 98 D8 98 D8
2040- B8 B8 B8 B8 0A 0A 0A 0A
2048- A8 E8 A8 E8 A8 E8 A8 E8
2050- A8 E8 A8 E8 0A 0A 0A 0A
2058- A8 E8 A8 E8 A8 E8 A8 E8
2060- B9 FD B8 F8 0A 0A 0A 0A
2068- B8 F8 B8 F8 B8 F8 B8 F8
2070- B9 FD 50 F8 0A 0A 0A 0A
2078- B8 F8 B8 F8 B8 F8 B8 F8
2080- B8 B8 C8 B8 0A 0A 0A 0A
2088- 48 28 48 28 48 28 48 28
2090- 48 28 C8 28 0A 0A 0A 0A
2098- 48 28 48 28 48 28 48 28
20A0- B8 B9 B8 B8 0A 0A 0A 0A
20A8- 58 38 58 38 58 38 58 38
20B0- 49 C9 58 38 0A 0A 0A 0A
20B8- 58 38 58 38 58 38 58 38
20C0- B8 B8 B8 B8 0A 0A 0A 0A
20C8- 68 08 68 18 68 08 68 18
20D0- 68 18 68 18 0A 0A 0A 0A
20D8- 68 08 68 18 68 08 68 18
20E0- 8D BD 78 70 0A 0A 0A 0A
20E8- 78 18 78 08 78 18 78 08
20F0- 0D 2D 78 70 0A 0A 0A 0A
20F8- 78 18 78 08 78 18 78 08

]LIST
    5 REM
   10 REM  PROGRAM TO MODEL DISK II
   20 REM  INTERFACE CARD. (MINI-
   30 REM  PROCESSOR ONLY.)
   40 REM
   50 REM  BY LEE MEADOR
   60 REM  COPYRIGHT (C) 1980
   70 REM
  100 B$="*":U$=" ": DIM FFD(6),FFQ(
      6),D(8),A(8),S(8),DL(8),A$(
      2),STS(30)
  105 DIM WRSTR$(100),HEX$(16):HEX$
      ="0123456789ABCDEF"
  107 POKE 2060,223
  110 FOR I=1 TO 6:FFD(I)=0:FFQ(I)
      =0: NEXT I
  115 FOR I=1 TO 30:STS(I)=0: NEXT
      I
  120 FOR I=1 TO 8:A(I)=0:D(I)=0:
      S(I)=0: NEXT I
  125 A(5)=1: REM  DEFAULT
  130 ROM=8192
  140 D$=": REM  ^D
  145 INPUT "PASCAL OR BASIC ",A$

  147 IF A$="P" THEN PRINT D$;"BLOAD D
      ISK ROM (PASCAL),A";ROM
  150 IF A$="B" THEN PRINT D$;"BLOAD D
      ISK ROM,A";ROM
  155 IF A$#"P" AND A$#"B" THEN END

 1000 REM  ***MAIN LOOP***
 1020 GOSUB 3000
 1030 GOSUB 4000
 1040 GOSUB 5000
 1050 GOTO 1000
 3000 REM  *** PRINT THE IN/OUT***
 3010 CALL -936
```

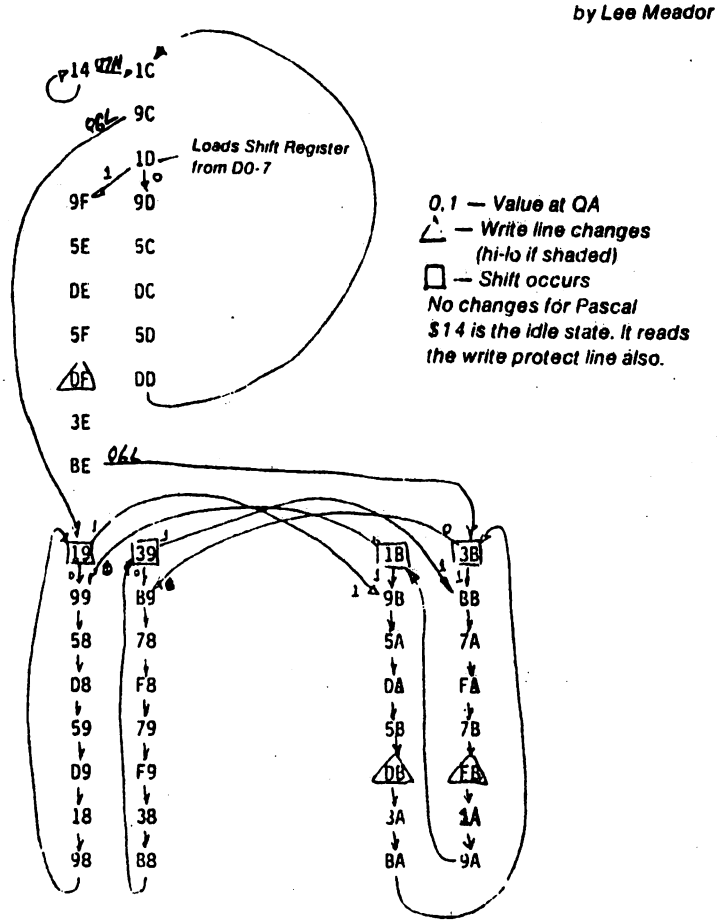
```
3020 PRINT
3030 PRINT "FF/D (0-5)  -) ";
3040 FOR I=1 TO 6: PRINT FFD(I);
     " ";: NEXT I: PRINT
3050 PRINT
3060 PRINT "FF/@ (1-5)  -) ";
3070 FOR I=1 TO 6: PRINT FF@(I);
     " ";: NEXT I: PRINT
3080 PRINT
3090 PRINT "@6 @7 @A SL SR S0 S1 WR R
     D CLR"
3100 PRINT " ";@6;"  ";@7;"  ";@A;
     "  ";SL;"  ";SR;"  ";S0;"  "
     ;S1;"  ";WRITE;"  ";READ;"  "
     ;CLR
3130 PRINT
3140 PRINT "ROM ADDRESS -) ";
3150 FOR I=8 TO 1 STEP -1: PRINT
     A(I);" ";: NEXT I: PRINT
3160 PRINT
3170 PRINT "ROM OUTPUT  -) ";
3180 FOR I=8 TO 1 STEP -1: PRINT
     D(I);" ";: NEXT I: PRINT
3190 PRINT
3200 PRINT "SHIFT REG   -) ";
3210 FOR I=8 TO 1 STEP -1: PRINT
     S(I);" ";: NEXT I: PRINT
3220 PRINT
3225 IF HDH<1 THEN HDH=1: IF HDL<
     1 THEN HDL=1
3230 PRINT "DATA BUS    -) ";HEX$
     (HDH,HDH);HEX$(HDL,HDL)
3240 PRINT
3300 REM ** GET NEW STATE **
3305 VAL=0
3310 FOR I=1 TO 8
3320 IF A(I) THEN VAL=VAL+2 ^ (I-
     1)
3330 NEXT I
3340 FOR I=1 TO 14
3350 STS(I)=STS(I+1): REM  KEEP 15 ST
     ATES
3360 NEXT I
3370 STS(15)=VAL
3375 REM ** PRINT 15 STATES BACK **
3380 FOR I=1 TO 15
3390 HI=STS(I)/16+1:LO=STS(I) MOD
     16+1
3400 VTAB I: TAB 35
3410 PRINT HEX$(HI,HI);HEX$(LO,LO)
     ;
3420 NEXT I
3500 REM ** DRAW WRITE LINE ****
3505 PRINT : PRINT
3510 IF NOT WRITE THEN WRSTR$( LEN(
     WRSTR$)+1)=U$
3520 IF WRITE THEN WRSTR$( LEN(WRSTR$
     )+1)=B$
3530 IF LEN(WRSTR$)>38 THEN WRSTR$
     =WRSTR$(2)
3540 PRINT WRSTR$;
3550 RETURN
4000 REM  *** LET CHANGES BE MADE**
4010 VTAB 20: TAB 1: CALL -958
4015 PRINT "YOU MAY CHANGE @6, @7, RD
      AND WR PROT"
4020 PRINT
4030 INPUT "TYPE 6,7,R OR W",A$
4045 IF A$="R" THEN READ= NOT READ
4060 IF A$="6" THEN @6= NOT @6
4070 IF A$="7" THEN @7= NOT @7
4073 IF A$="W" THEN SR= NOT SR
4090 IF A$="" THEN RETURN
4100 GOTO 4010: REM  MORE CHANGES
5000 REM   *** DO ONE CLOCK CYCLE***
5010 GOSUB 6000: REM  FLIP FLOP
5020 GOSUB 8000: REM  SHIFT REG
5030 GOSUB 7000: REM  ROM
5040 RETURN
6000 REM ***SET UP FF AND CLOCK IT***

6001 FFD(1)=D(8): REM  SET DATA IN
6002 FFD(2)=D(7)
6003 FFD(3)=D(5)
6004 FFD(4)=FF@(5)
6005 FFD(5)=READ
6006 FFD(6)=D(6)
6009 REM ***CLOCK IT ACROSS***
6010 FOR I=1 TO 6:FF@(I)=FFD(I):
      NEXT I
6020 RETURN
7000 REM ***SET UP ROM ADDR, GET DATA
     ***
7005 A(1)=FF@(6): REM  SET THE ADDR L
     INE
7010 A(2)=@A
7020 A(3)=@6
7030 A(4)=@7
7040 A(5)= NOT (FF@(4) AND ( NOT
     FF@(5)))
7050 A(6)=FF@(3)
7060 A(7)=FF@(2)
7070 A(8)=FF@(1)
7075 X=A(6):A(6)=A(8):A(8)=X
7080 WRITE=FF@(1): REM   SET WRITE
7090 OFFSET=0
7095 REM ***LOOK INTO ROM FOR DATA***

7100 FOR I=1 TO 8
7110 IF A(I) THEN OFFSET=OFFSET+
     2 ^ (I-1)
7120 NEXT I
7130 VAL= PEEK (ROM+OFFSET)
7140 FOR I=1 TO 8
7150 D(I)=VAL MOD 2
7160 VAL=VAL/2
7170 NEXT I
7171 REM ***SWAP SINCE DATA LINES ARE
      SWAPPED ON THE P5 ROM***
7172 X=D(8):D(8)=D(5):D(5)=X
7174 X=D(7):D(7)=D(6):D(6)=X
7200 RETURN
8000 SL=D(3): REM  SET OUTPUTS
8010 S1=D(1)
8020 S0=D(2)
8030 CLR=D(4)
8080 IF NOT CLR THEN 8270
8090 IF NOT (S1 OR S0) THEN RETURN

8100 IF S1 AND S0 THEN 8400
8120 IF S0 AND ( NOT S1) THEN 8200

8130 IF NOT (S1 AND ( NOT S0)) THEN
     RETURN
8140 REM   **SHIFT LEFT***
8150 FOR I=8 TO 2 STEP -1
8160 S(I)=S(I-1)
8170 NEXT I
8180 S(1)=SL
8190 GOTO 8600
8200 REM   **SHIFT RIGHT**
8210 FOR I=1 TO 7
8220 S(I)=S(I+1)
8230 NEXT I
8240 S(8)=SR
8250 GOTO 8600
8270 REM **CLEAR SHIFT REG**
8280 FOR I=1 TO 8
8290 S(I)=0
8300 NEXT I
8310 GOTO 8600
8400 REM ***LOAD SHIFT REG***
8410 INPUT "LOADING SHIFT REG. ENTER
     HEX $",A$: IF LEN(A$)<1 THEN
     8410
8420 D1= ASC(A$):D2= ASC("0"): IF
     LEN(A$)>1 THEN D2= ASC(A$(2
     ))
8430 D1=D1- ASC("0"):D2=D2- ASC(
     "0")
8440 IF D1>9 THEN D1=D1-7: IF D2>
     9 THEN D2=D2-7
8442 IF LEN(A$)=1 THEN DTA=D1
8444 IF LEN(A$)>1 THEN DTA=D1*16
     +D2
8450 IF D1<0 OR D1>15 THEN 8410:
      IF D2<0 OR D2>15 THEN 8410

8470 FOR I=1 TO 8
8480 DL(I)=DTA MOD 2
8490 DTA=DTA/2
8495 S(I)=DL(I)
8500 NEXT I
8540 GOTO 8600
```

```
8600 REM ***FINISH SETTING OUTPUTS!

8605 HDH=0:HDL=0
8610 QA=S(8)
8620 FOR I=1 TO 4
8630 DL(I)=S(I)
8640 IF DL(I) THEN HDL=HDL+2 ^ (
     I-1)
8650 NEXT I
8660 FOR I=5 TO 8
8670 DL(I)=S(I)
8680 IF DL(I) THEN HDH=HDH+2 ^ (
     I-5)
8690 NEXT I
8695 DTA=HDH*16+HDL
8696 HDL=HDL+1:HDH=HDH+1
8700 RETURN
```

```
FF/D (0-5)   -> 0 0 0 0 0 1

FF/Q (1-5)   -> 0 0 0 0 0 1

Q6 Q7 QA SL SR S0 S1 WR RD CLR
 0  0  0  0  0  0  0  0  0  1

ROM ADDRESS -> 0 0 0 1 0 0 0 1

ROM OUTPUT  -> 0 0 1 1 1 0 0 0

SHIFT REG   -> 0 0 0 0 0 0 0 0

DATA BUS    -> 00

    YOU MAY CHANGE Q6, Q7, RD AND WR PROT

TYPE 6,7,R OR W
```

Disk Card Mini-Processor States (WRITING)

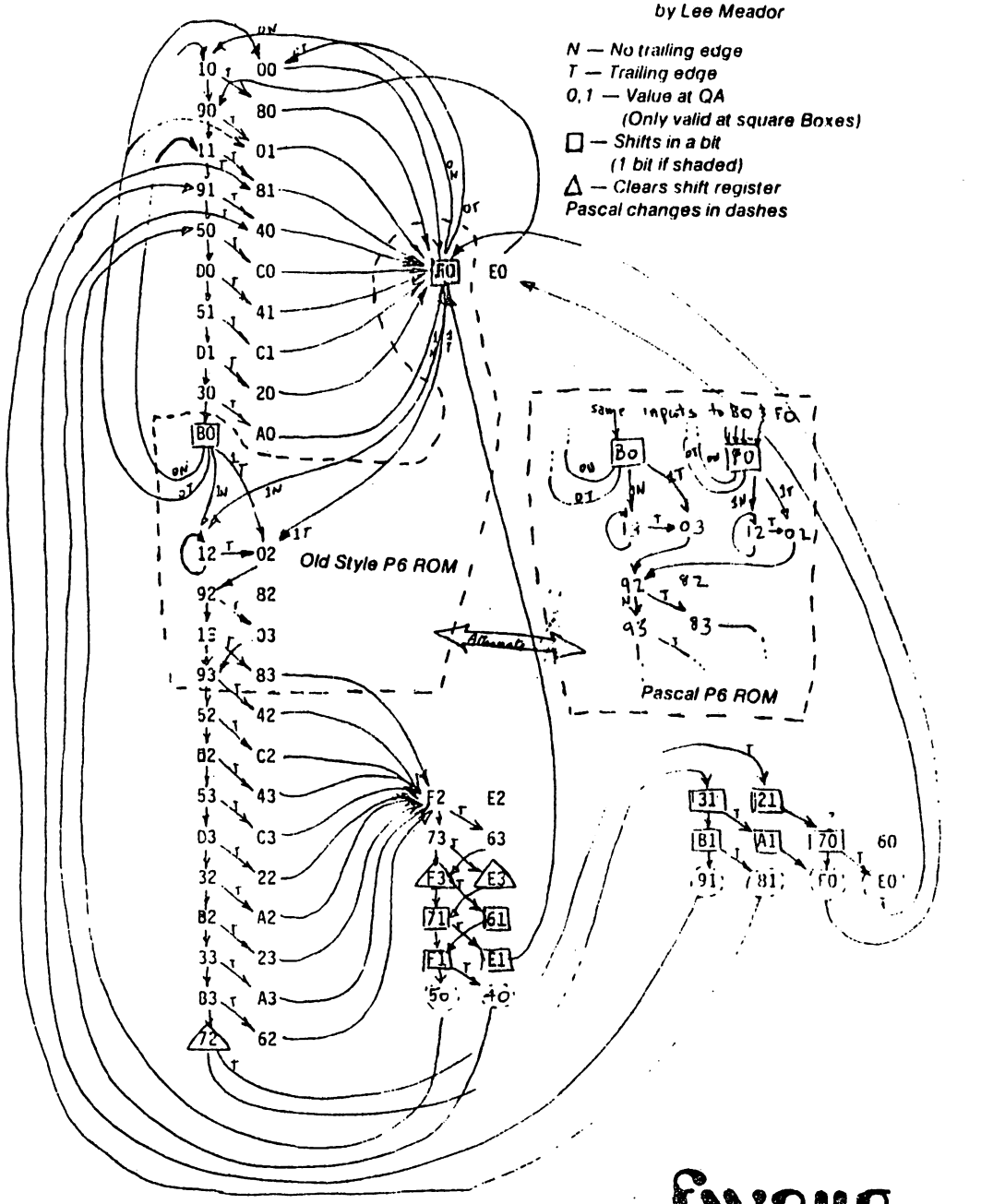
by Lee Meador



Disk Card Mini-Processor States (READING)

by Lee Meador

N — No trailing edge
T — Trailing edge
0,1 — Value at QA
(Only valid at square Boxes)
□ — Shifts in a bit
(1 bit if shaded)
△ — Clears shift register
Pascal changes in dashes



fwaug

<<< WANT AND DON'T WANT ADS >>>

2 DI/AN PRINTERS. Used and for sale in good condition with I/O device and software. RS-232. Lewis Melton, 981-8866.

FOR SALE: HEATH H-14 Dot Matrix Printer RS232 or current loop interface, high speed, forms control, three print sizes.....$900 + tax at Heath. Will sell for $500. See Mike Kramer or call at 358-6687 after 5.

FOR SALE: Apple Integer Card $150. See Mike Kramer or call 358-6687 after 5.

SANYO MONITORS AVAILABLE IN GROUP PURCHASE. We need a minimum of 6 ordered if we are to get the special prices.

13" color	$430.	+ tax	(30-day delivery)
9" B & W	169.	"	(stock)
12" B & W	200.	"	(8-10 days delivery)
15" B & W	250.	"	(stock)

If you are interested contact Ray Essig, 493-9980 or 497-7165 (evenings).

BACK ISSUES OF APPLE BARREL are for sale in limited quantities! Many of you have inquired about their availability. The following back issues can be bought by mail for $1.00 each, postpaid:

vol. 2 no. 5	August, '79
vol. 2 no. 6	Sept/Oct, '79
vol. 3 no. 1	January, '80
vol. 3 no. 2	February, '80
vol. 3 no. 3	Mar/Apr, '80
vol. 6 no. 6	August, '80
vol. 7 no. 7	Sept/Oct, '80
vol. 8 no. 8	November, '80

This is a chance for newer members of HAAUG to catch up on programs, news, reviews, etc. Sorry, but there will be NO reprints when these are gone. Make checks payable to H.A.A.U.G. and send to Apple Barrel; Ed Seeger, Editor; 4331 Nenana Drive; Houston, TX; 77035. Please allow 2 weeks for delivery.

SUPER.TEXT WORD PROCESSOR, version 2, by Muse for sale at $85 in mint condition. This is one of the "big two" (EasyWriter is the other) implemented on the Apple in the $100 range. Worth $100 if you wish to trade it in for their Super.Text II system at $150. Has math mode (!), built-in copy routine for files, and displays upper & lower case ON SCREEN with Paymar chip. Ed Seeger evenings at 723-6919.

Dear Dr. Apple:

The sales clerk explained the situation to me when I bought my new Apple, but in the joy of a new toy I forgot. What are the language capabilities within the Apple computer and its augmentations? I have an Apple II Plus.

Signed,
Coming Down to Earth

Dear Down to Earth:

You are reminded that both the Apple II and the Apple II Plus computers are identical except for the basic language ROM chips built into them.

Apple II computers have Integer Basic burned into them at the factory. A second language, Applesoft Basic, is available to the Apple II through a cassette conversion program which used to come free with the computer purchase. For those with disk drives, the 3.2 DOS Master Diskette has included a copy of the same Applesoft conversion program. One has to load and run the conversion program before operating any program written in the Applesoft language. Since this is a bit awkward and time consuming, Apple manufactures a firmware Applesoft card which provides instantaneous transition in and out of Applesoft when ever required. Pascal language can be added to the Apple through purchase of the Apple Language RAM card, PROM chips and Pascal compiling program diskettes, all of which requires the user to have at least one disk drive to start with. Additionally as a bonus, the Pascal System has an accompanying diskette free which provides for program operations in Applesoft. Lastly, Fortran language capability can be added to the above system through two additional diskettes. Moreover, COBAL and other language capabilities are planned to come into the Apple the same way.

Now, Apple II Plus computers have Applesoft Basic manufactured into them to begin with, but no Integer. There are no Integer cassete conversion program available at this time, however there is an Integer Basic conversion program on diskette just now coming on the market which permits operation of Integer programs on the computer. A firmware Integer card (counterpart to the firmware Applesoft card mentioned above) is available for direct electronic transition into that language when inserted into the Apple II Plus. Of course Pascal can be obtained through the same Pascal System discussed earlier and the same accompanying diskette brings in Integer as its extra gift to Apple II Plus users.

Incidentally, the Assembly language of both computers is exposed through the system Monitor. The table below gives language availabilities at a glance:

Language Source	Apple II	Apple II Plus	Remarks
Built into Computer	Integer & Assembly	Applesoft & Assembly	ROM
Cassette	Applesoft[1]	None yet	12k RAM used
Diskette	Applesoft[1]	Integer[2]	12k RAM used
Firmware Card[3]	Integer<->Applesoft	Applesoft<->Integer	Automatic
Language System[4]	Pascl,Aplsft,Fortrn[5]	Pascl,Intgr,Fortrn[5]	Need disk drive

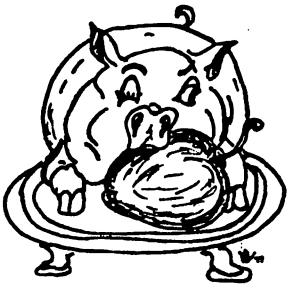
Note 1. Free, HAAUG Library
Note 2. Retail $20
Note 3. Retail $200
Note 4. Retail $500
Note 5. Retail $200 extra

Signed,
Dr. Apple

Houston Area Apple Users Group
APPLE BARREL
Ed Seeger, Editor
4331 Nenana Drive
Houston, Texas 77035

(713) 723-6919

BULK RATE
U.S. POSTAGE
PAID
HOUSTON, TEXAS
PERMIT 3936



H.A.A.U.G

Postmasters:

Address Correction Requested:
Forwarding and Return Postage Guaranteed