

\$1

# Washington Apple Pi



Volume 1

December 1979

Number 11

## Highlights

**Apple Memory Atlas pg. 18**

**Apple Calculator pg. 5**

## In This Issue

	Page
Event Queue	1
Classified Ads	1
Editor's REMs	1
Minutes	1
Ampersort - Alan G. Hill	2
Auto Number - Gerald Cahill	3
In My Opinion - Sandy Greenfarb	4
Adding a Calculator to your Apple - Paul Sand	5
Product Review: Softside/Apple Seed - Sandy Greenfarb	6
Page List for the Apple - Robert D. Diaz	7
Lowers - Paul Rinaldo	9
Adding Colors to Apple II Hi-Res	10
Apple User Groups	12
Parallel Interface - Dave Skillman	16
Bytes from the Apple Pi - Sandy Greenfarb	16
Adjusting the Disk II Speed - Mark L. Crosby	17
What's Where in the Apple - Prof. William F. Luebbert	18
Memory from 0 to 65535 - Cider Press	22
Graphics Driver for the IDS 440 Printer - Hersch Pilloff	23

# NEW APPLE II® SOFTWARE

PROGRAMMA Software Program Products



FUNCTION PLOT \$24.95



APPLE INVADERS GAME



I-CHING \$15.95



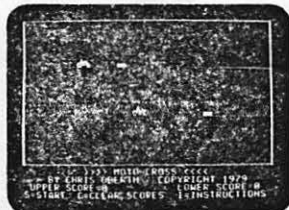
TRIVIA BOX \$19.95



CASSETTE \$15.95



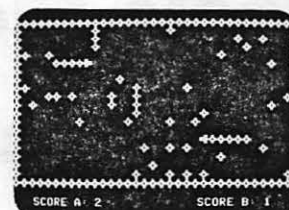
SHAPE BUILDER \$19.95



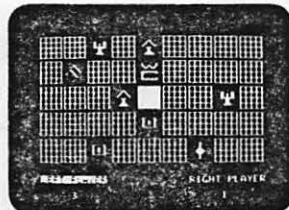
MOTO-CROSS \$9.95



DISKETTE \$19.95



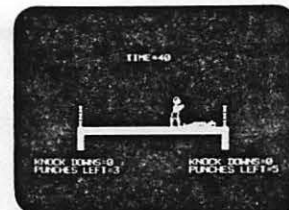
BLOCKADE \$9.95



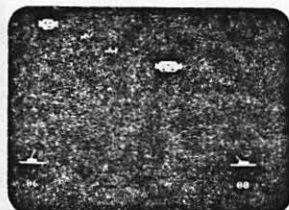
FRUSTRATION \$9.95

## AND MORE...

ACTIVE FILTERS	\$24.95
ALIEN INVASION	9.95
AMPERSORT II	15.95
APPLE ALLEY	6.95
BASEBALL	15.95
BATTLEFIELD	9.95
BREAKTHRU	9.95
CHECK BOOK	34.95
DATABASE MAILER	29.95
DEATH RACE	15.95
EARTH QUEST	19.95
HOME BUDGET	24.95
HOUSEHOLD FINANCE	24.95
MINI GENERAL LEDGER	59.95
MOUSE HOLE	6.95
PEG JUMP	9.95
RICOCLETTE	9.95
STAR VOYAGER	15.95
STUNT CYCLE	15.95



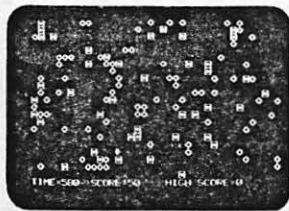
BOXING \$9.95



GUIDED MISSILE \$15.95



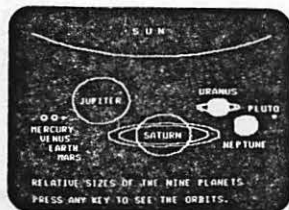
ALGEBRA I \$15.95



LASER BLAST \$9.95

All orders must include 3% postage and handling. California residents add 6% sales tax. VISA and MASTERCARD accepted.

Apple II is a trademark of Apple Computers, Inc.



THE PLANETS \$15.95

**PROGRAMMA INTERNATIONAL, Inc.**  
 3400 Wilshire Blvd.  
 Los Angeles, CA 90010  
 (213) 384-0579  
 384-1116  
 384-1117

Dealer Inquiries Invited



SIRIUS \$15.95

## 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

## EVENT QUEUE

The next meeting of the Washington Apple Pi is December 15, 1979 at 9:30 am at G.W. University School of Engineering in Tompkins Hall Room 206 which is located at the corner of 23rd and H streets N.W. NOVAPPLE will continue to meet on second Wednesdays at Computers Plus, Franconia and fourth Thursdays at Computerland of Tysons Corner.... for December, the 12th at Computers Plus; that will be the last meeting of 1979. January 10th will be the first meeting in 1980, also at Computers Plus followed by January 25th at Computerland.

## Classifieds

FOR SALE - Trendcom Printer and Apple Interface, \$400, payments can be arranged. Jim Manley, days-(202) 261-8250 Ext-1471, eves (301) 426-9248. This is the 40 column NCR blue or black sensitized paper model.

Christmas special - Last Chance. Novation Modem model 3102A, Apple Communications Interface Card; \$175 each. George Hinds (301) 585-0979

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.

## EDITOR'S REMS

I want to apologize for the size of the last issue of the Washington Apple Pi. Because of the lack of articles and personal commitments, we just couldn't do much better. We have, however, made up for that in this issue - as you will soon find out. Please read Sandy Greenfarb's unsolicited editorial for some astute observations about our club. Then we have many excellent programs such as adding a calculator program to your Apple, adding a page list to aid your programming and more. Also included is a complete memory atlas and the final version of Alan Hill's "Ampersort" for fast array sorting using the "&" command. We have also compiled a complete (at least for a few days anyway) Apple Users Group listing from all over the world.

ONE IMPORTANT NOTE: Please fill out completely the new Washington Apple Pi Membership Application Form if you wish to join the Washington Apple Pi. We also have an order form for a group purchase of diskettes and a program library for sale! (Aplause, applause)

Due to an error, we inadvertently left out the NOVAPPLE membership application form that was mentioned in last month's NOVAPPLE minutes. This month we have included it. Anyone who joined Apple Pi but meant to join NOVAPPLE need only notify the Apple Pi officers and we will transfer your membership to NOVAPPLE. Of course, there is nothing wrong with belonging to both clubs for the best of both possible worlds.

mark

## MINUTES

NOVAPPLE minutes of 14 November 1979: The meeting was called to order at 7:45 pm by the President. He announced that the next meeting would be held on the 29th of November due to Thanksgiving conflicts. A brief discussion was held concerning the printing of a membership list. It was brought out that there are several problems regarding this but it was agreed that further discussion would be solicited at another meeting. The discussion was tabled until more persons would have a chance to respond. A motion was made to set up a tutorial program for other aspects of the Apple similar to the machine language program being offered by Kim Woodward. The motion was approved. (In order to have such a program we need someone to volunteer his or her services to plan a program. We will assist any person who is willing to attempt the tutorial with view graph aids, and any other help they might need.) Please let the officers know if you can perform this service. A program was presented for the evening which presented a game expansion I/O scheme and a joystick which can be used for the Apple. The meeting broke up around 9:00.

NOVAPPLE minutes of 29 November 1979: The meeting was called to order at 7:45 pm by the President. Because the next meeting would come during the Christmas holidays, the consensus of the members attending was that the meeting for December 12, 1979 at Computers Plus should be the last meeting for the year. The next meeting in January would be the 10th of January at Computers Plus and 25 January at Computerland of Tysons Corner. The program was again presented by Kim Woodward. He continued his fine presentation of machine language programming. Using view graphs and the Apple itself he reviewed the past material presented and took us further into the mysterious world of machine language. We look forward to a few more lessons before he is done. For those of you who have not paid your dues to NOVAPPLE for the next six months, they were due in October. An application form has been included in this issue of Apple Pi to aid you in becoming current. The dues are \$6.00 for six months. Make your checks payable to Northern Virginia Apple Users Group and send to NOVAPPLE, PO Box 10411, Alexandria, Virginia 22310. Please fill out the application form in full as we need the information for our records.

Gerald Eskelund, Secretary

# AMPERSORT

A fast, machine language sort utility for the APPLE II that handles integer, floating point and character records. Because it is callable from BASIC, this sort routine is a worthwhile addition to any software library.

A sort utility is usually one of the first programs needed for records management application programs. If the utility is written in BASIC and runs under an interpreter, one quickly discovers that the sort is painfully slow on a micro. The sort program presented here, written in machine language for the APPLE II with AppleSoft ROM, will certainly remedy that problem. While no speed records will be set, it will run circles around BASIC, sorting 900 integer, 700 floating point, or 300 30-character records in about 60 seconds.

Speed is not the only beauty of AMPERSORT. As its name implies, the BASIC-to-machine language interface utilizes the powerful, but not widely known, feature of AppleSoft — the Ampersand. What is the Ampersand and why is it so useful? Consider the following example of how a BASIC program passes sort parameters to AMPERSORT:

```
100 &SORT#(AB$,0,10,7,10,A,1,5,D)
```

This statement when embedded in a BASIC program or entered as an immediate command, will command AMPERSORT to sort AB\$(10) in ascending order based on the 7th to 10th characters and in descending order for the 1st through 5th characters. Of course, POKEs could be used to pass parameters from other 6502 BASICS, but there's something more professionally pleasing about the Ampersand interface.

There is no user documentation from APPLE on the Ampersand feature. I first read of the feature in the October 1978 issue of CALL APPLE. When the AppleSoft interpreter encounters an ampersand (&) character at the beginning of a BASIC statement, it does a JSR \$3F5. If the user has placed a JMP instruction there, a link is made to the user's machine language routine. APPLE has thoughtfully provided some ampersand handling routines described in the November and December issues of CALL APPLE. The routines enable your machine language routine to examine and convert the characters or expressions following the ampersand. The routines used in AMPERSORT are:

## CHRGET (\$00B1)

This routine will return, in the accumulator, the next character in the statement.

The first character is in the accumulator when the JSR \$3F5 occurs. The zero flag is set if the character is an end-of-line token (00) or statement terminator (\$3A). The carry flag is set if the character is non-numeric, and cleared if it is numeric. The character pointer at \$B8 and \$B9 is advanced automatically so that the next JSR \$B1 will return the next character. A JSR \$B7 will return a character without advancing the pointer.

## FRMNUM (\$DD67)

This routine evaluates an expression of variables and constants in the ampersand statement from the current pointer to the next comma. The result is placed in the floating point accumulator.

## GETADR (\$E752)

This routine will convert the floating point accumulator to a two-byte integer and place it in \$50 and \$51. FRMNUM and GETADR are used by AMPERSORT to retrieve the sort parameters and convert each to an integer.

## GETBYT (\$E6F8)

This routine will retrieve the next expression and return it as a one-byte integer in the X-register.

It is the user's responsibility to leave the \$B8 and \$B9 pointer at the terminator.

Parameters are passed to AMPERSORT in the following form:

```
100 &SORT#(AB$,B,E,7,10,A,1,5,D)
```

where:

AB\$ is the variable name of the string array to be sorted. The general form is XX\$ for string arrays, XX% for integer arrays, and XX for floating point arrays.

B is a variable, constant or expression containing the value of the subscript element where the sort is to begin, e.g. AB\$(B).

E is a variable or constant or expression containing the value of the subscript element where the sort is to end, e.g. AB\$(E) B and

Alan G. Hill  
12092 Deerborn Drive  
Cincinnati, OH 45240

E are useful when the AB\$ array is partially filled or has been sectioned into logically separate blocks that need to be sorted independently.

7 is a variable, constant or expression specifying the beginning position of the major sort field.

10 is a variable, constant or expression specifying the ending position of the major sort field.

A is a character specifying that the major sort field is to be sorted in ascending order.

1 is a variable, constant or expression specifying the beginning position of the first minor sort field.

5 is a variable, constant or expression specifying the ending position of the first minor sort field.

D is a character specifying that the first minor sort field is to be sorted in descending order.

The &SORT command will sort character, integer or floating point arrays and can be used in either the immediate or deferred execution mode similar to other AppleSoft BASIC commands. Of course, the named array must have been previously dimensioned and initialized in either case.

## A Character Arrays

1. Equal or unequal element lengths
2. Some or all elements
3. Ascending or descending order
4. A major sort field and up to 4 minor sort fields

Examples:

```
10 DIM NA$(500)
```

```
100 &SORT#(NA$,0,500,1,5,A)
200 &SORT#(NA$,0,500,1,5,A,6,10,
    D,11,11,A)
299 F% = 0: L = 10
300 &SORT = (NA$,F%,L,10,15,D)
```

Line 100 sorts on positions 1 through 5 in ascending order for all 501 elements of NA\$(500)

Line 200 is the same as Line 100 except that minor sort fields are specified. The sort sequence on positions 1-5 is in ascending order, positions 6-10 are in descending order, and position 11 is ascending order.

Line 299 and 300 sort on positions 10-15 in descending order for NA\$(0) through NA\$(10).

## B. Integer and Floating Point Arrays

1. Some or all elements
2. Ascending order only. (Step through the array backwards if needed in descending order.)

Examples:

```
10 DIM AB%(100),FP(100)
```

```
100 &SORT#(AB%,0,100)
299 S = 50: E = 100
300 &SORT#(AB%,S,E)
399 X = 49
400 &SORT#(FP,0,X)
```

Line 100 sorts all 101 elements of AB%(100) in ascending order. Lines 299 and 300 sort from AB%(50) through AB%(100), while lines 399 and 400 sort from FP(0) through FP(49).

Limited editing has been included in the parameter processing code. Therefore, one must be careful to observe such rules as:

1. 0<B<E<< maximum number of AB\$ elements.
2. AB\$ must be a scalar array, e.g. AB\$(10), not AB\$(20,40).
3. The sort array name must be less than 16 characters only the first two count, and they must be unique.
4. The maximum number of sort fields is 5.
5. The beginning sort field position must not be greater than the ending sort field position.

Options:

1. Constants, variables, or expressions may be used for subscript bounds and sort positions.
2. The &SORT command may be used in immediate or deferred execution mode.

Some editing checks are made. You will notice this when you get a "SYNTAX ERROR IN LINE XXX" error message. You will also get a "VARIABLE XXX NOT FOUND" message if the routine cannot find the AB\$ variable name in variable space.

The AMPERSORT program is listed in its entirety. A BASIC demo program is also shown. Anyone desiring a cassette tape containing the latest version of the object code assembled at \$5200, a copy assembled at \$9200, and the source program text in the Microproducts APPLE II Assembler format may receive these by sending the author \$5.00 at the above address.

14:40  
MICRO—The 6502 Journal  
July 1979

## AMPERSORT Demo

```
1000 GOTO 10000
1050 REM CHARACTER SORT
1060 CH$ = "ABCDWXYZ": L = LEN (CH$) - 1
1070 NX = 8
1080 DIM AB$(NX)
1090 FOR I = 0 TO NX
1100 C$ = MID$(CH$, INT ( RND (1) * L) + 1, 1)
1110 B$ = MID$(CH$, INT ( RND (1) * L) + 1, 1)
1120 FOR J = 1 TO 3
1130 C$ = C$ + C$: B$ = B$ + B$
1140 NEXT J
1150 AB$(I) = B$ + C$
1160 NEXT I
1170 GOSUB 1240
1180 REM SORT HALF ASCENDING
1190 REM SORT HALF DESCENDING
1200 & SORT#(AB$,0,NX,1,8,A,9,16,D)
1210 GOSUB 1260
1220 GOTO 11000
1230 REM PRINT ROUTINE
1240 PRINT " BEFORE"
1250 GOTO 1270
1260 PRINT " AFTER": PRINT "ASCEND DESCEND"
1270 FOR I = 0 TO NX
1280 PRINT AB$(I): NEXT I: RETURN
2000 REM INTEGER SORT
2010 NX = 8
2020 DIM IN$(NX)
2030 FOR I = 0 TO NX
2040 IN$(I) = 7500 - INT ( RND (1) * 15000)
2050 NEXT I
2060 GOSUB 2120
2070 REM SORT
2080 & SORT#(IN$,0,NX)
2090 GOSUB 2130
2100 GOTO 11000
2110 REM PRINT ROUTINE
2120 HTAB 10: PRINT "BEFORE": GOTO 2140
2130 HTAB 10: PRINT "AFTER"
2140 FOR I = 0 TO NX
2150 PRINT IN$(I): NEXT I: RETURN
3000 REM FLOATING POINT
3010 TX = 8
3020 DIM FP(TX)
3030 FOR I = 0 TO 8
3040 FP(I) = 1000 * RND (1) * SIN (I * 7.16)
3050 NEXT I
3060 GOSUB 3120
3070 REM SORT
3080 & SORT#(FP,0,TX)
3090 GOSUB 3130
3100 GOTO 11000
3110 REM PRINT ROUTINE
3120 HTAB 10: PRINT "BEFORE": GOTO 3140
3130 HTAB 10: PRINT "AFTER"
3140 FOR I = 0 TO TX
3150 PRINT FP(I): NEXT I: RETURN
10000 REM ** &SORT DEMO **
10010 REM SAVE ROOM FOR
10020 REM SORT ROUTINE
10030 HIHEX: 20992: REM $5200
10040 D$ = CHR$(4)
10050 PRINT D$;"BLOAD B.AMPER-SORT"
10060 REM SET UP 'A' HOOK
10070 REM AT $3F5:JMP $5200
10080 POKe 1013,76: POKe 1014,0: POKe 1015,82
10090 HOME : CLEAR
10100 UTAB 8: HTAB 15: PRINT "SORT DEMO"
10110 PRINT : HTAB 15: PRINT "SELECTIONS"
10120 PRINT : HTAB 10: PRINT "1 INTEGER SORT"
10130 HTAB 10: PRINT "2 FLOATING POINT SORT"
10140 HTAB 10: PRINT "3 CHARACTER SORT"
10150 HTAB 10: PRINT "4 EXIT"
10160 UTAB 17: INPUT "SELECTION #ISEX
10170 IF SEX < 0 OR SEX > 4 THEN 10090
10180 ON SEX GOTO 2000,3000,1050,1010,10190
10190 END
11000 PRINT "HIT ANY KEY TO RETURN TO MENU"
11010 WAIT - 16384,128
11020 POKe - 16368,0
11030 GOTO 10090
```

(CONTINUED)....





# ComputerLand<sup>®</sup> and apple II

For the best in personal computing

**SOFTAPE**  <sup>TM</sup>

**Personal Software** <sup>TM</sup>

**D. C. Hayes Associates, Inc.**  
MICROCOMPUTER PRODUCTS

**CENTRONICS** <sup>®</sup>

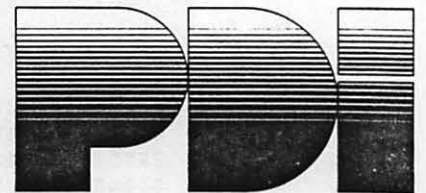
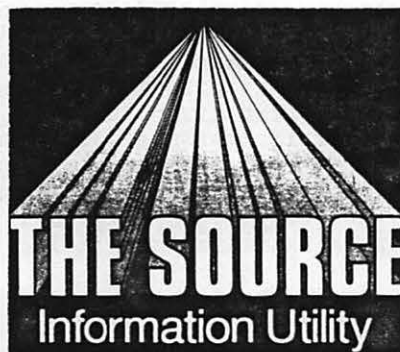


**Mountain Hardware, Inc.**



**Integral Data Systems, Inc.**

**MUSE**



**houston  
instrument**

**Heuristics**  
INC.



**Automated Simulations**

**SANYO**



**ComputerLand** <sup>®</sup>

**We Know Small Computers.**

ComputerLand/Tyson's Corner  
8411 Old Courthouse Road at Rt. 123 — 893-0424



# In my Opinion...

Sandy Greenfarb

Just over a year ago, a group of APPLE II owners and users gathered in College Park in what might be loosely termed as a meeting. Even though the name was not to originate for several more months, this was technically the genesis for Washington Apple Pi.

Quite a lot has happened in a year. The "Pi" has grown to a solid and substantial membership. A few monthly mimeographed sheets have turned into a high-quality magazine. The club has even acquired a large library of public domain programs for the use of the members. The size alone of our monthly turnouts would lead one to believe that our club was a thriving success. But are we?

My personal opinion is that we have reached a critical point in the life of our club and, by virtue of our actions, can either go up or down. Standing still is just not possible. Let us take another look at our "success". Our magazine exists mainly through the exhausting efforts of three individuals, several steady contributors, and occasional others. Our library exists simply because several of our members have purchased other club libraries and integrated them into our own. Quite tragically, almost none of the programs in our library were originated by members. Thirdly, as a question proposed to the reader, I ask, "What portion of the attending members have come to the meetings to share or contribute and what portion do you feel has come to get something for nothing?"

4 If I've insulted you by now, then at least I've got you thinking. At this time, I'll soften the blow and admit that I was only trying to convey one possible impression as to what is happening and that you were deliberately guided to this wrong impression. Below the surface I feel that things are slightly different. We have many budding authors with beneficial facts and articles to share with other members, but they have over-humbled themselves. They might feel that what they have to contribute is too minor or too basic. They might feel that it is audacious of them to present material to our "so called" experts. They might also be wrong on every account, if that's the way they feel. Our preamble contains the phrase "mutual learning and education". By definition we share our efforts with others. I consider myself somewhat of an expert on some aspects of APPLE II programming, yet I greedily absorb every printed word on the subject. The more I learn, the more I see there is to learn. I do not say this on an ego trip, but rather to point out that I do read articles written by people less experienced and I do learn from them and I am grateful that they were written. If you have something to write about which would be informative to others, then write it and submit it to our editor. Don't even worry about its length. I, with my tendency to be verbose, would envy you for saying in a paragraph what takes me a full page.

My second point is the club library. Again, I emphasize not to be too humble. If you have original software that you would be willing to contribute, then contact the librarian. Maybe you enhanced something that already existed or just copied something out of a magazine. Contact him. Probably someone else will want the same program. Possibly you have saved another member from paralleling your effort and that individual, in turn, will instead work on another program of benefit to you. This is what "mutual" is all about.

Third, let me degrade my rhetorical question about cooperating members. I blame myself in part for this as I feel that several of us have presented the impression of being "experts" and have scared valued contributors into keeping quiet. I also blame a lack of direction in some of our meetings.

All in all, I consider our first year successful, but back to the critical point. We have two choices. We can stay at our present level which I feel will lead us to stagnation and downhill, or we can add new goals. I propose the latter and make the following recommendations:

Appoint an agenda committee. Too much of our meetings is hit or miss. A lot of people have beneficial things to say with no appropriate place to fit them in. Too much of our present format is business. One possible format is: business (with a maximum time limit), planned subject matter, and general discussion. In this aspect, the membership can set the guidelines for what is desired for planned subject matter, i.e., demonstrations, subject lectures and/or seminars.

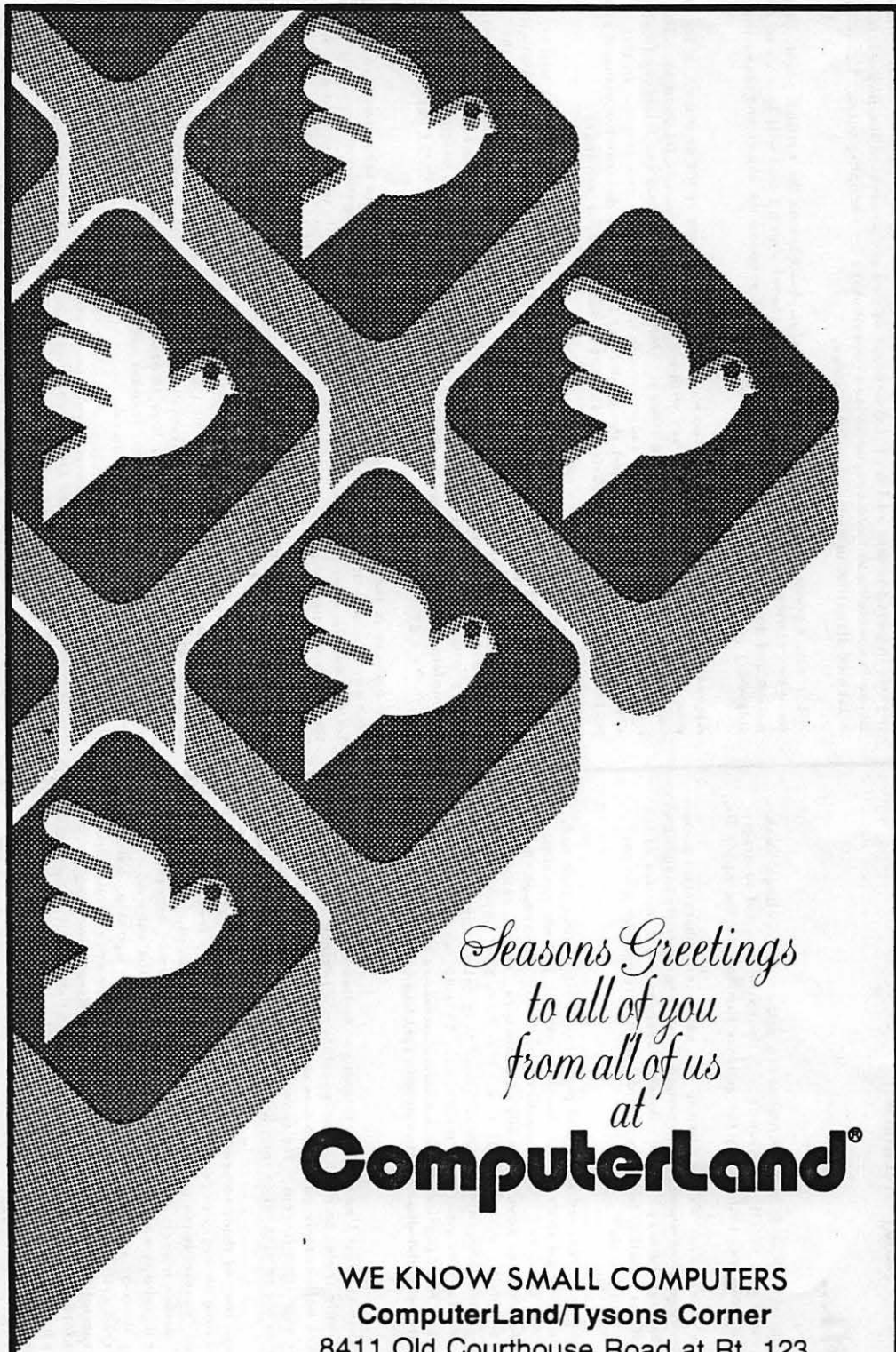
Actively pursue special interest groups. I for one am a gamester and would be quite willing to join with others in creating new games for the APPLE. (Actually, I'm at fault for not advertising this fact in our magazine.) I know of others that prefer business or education applications. Nonetheless, it's time to band the special interests and derive group efforts that easily outproduce individual efforts. Let's start pooling our respective talents. I've always been weak on originality but strong on implementation and enhancement. Somewhere in this club is my opposite and we can make a great team.

Buy a share in the club. Despite surface appearance, it does take money to run the club and the club does need money "up front" for putting together the monthly newsletter, supplies for the library, etc. I propose, subject to membership approval, that each dues paying member buy a \$10 share in the club. Said money would be refunded if a member quits or fails to renew annual membership. One benefit would be to maintain a minimum balance and avoid check charges. I'm sure the treasurer could present other benefits.

Don't forget the "newbee". We were all novices once. One inherent problem in a technical club, such as ours might be considered to be, is that the group tends to advance, and at the same time advances its skill level. New members find themselves left ignorantly in the cold, lose interest and quit. Unless the club desires to become a closed, stagnated group, it must provide a broad enough forum for all members, from beginner to advanced.

CONTRIBUTE! Give your time, knowledge, money, services, whatever. Consider what you feel you have gained from being a member and whether or not you are indebted in some form or fashion. If the club has given something beneficial to you, maybe you can do something in return, from volunteering to lick the stamps for the monthly mailing to writing a treatise on the use of Page Zero locations \$3C thru \$43.

In the past year we have grown from conception to an outstanding club. Including our companion club, NOVAPPLE, I feel that, with the exception of the suburbs of Silicon Valley (which beat us simply on quantity), our APPLE group is one of, if not the most active of all. We are blending well and doing well together. A lot has been accomplished in our first year, but year 2 is upon us. Let's not rest on laurels.



*Seasons Greetings  
to all of you  
from all of us  
at*

**ComputerLand<sup>®</sup>**

WE KNOW SMALL COMPUTERS  
ComputerLand/Tyson's Corner  
8411 Old Courthouse Road at Rt. 123  
893-0424

 **apple computer**  
Sales and Service



# ADDING A CALCULATOR TO YOUR APPLE

Paul Sand

Chances are that your powerful Apple computer has not yet replaced your \$19.95 pocket calculator. This calculator emulation program written in Applesoft may reduce, if not eliminate, your need for a calculator when your Apple is handy.

This program incorporates a subset of the functions found on a "scientific" calculator (sine, cosine, logarithm, etc.). It uses "Reverse-Polish Notation" (RPN) and a "stack" to store the numbers currently in use. (If either term is unfamiliar, don't worry - use of the program illustrates the concepts.)

When running, the program will accept a (real) number or a "command" (terminated by pressing RETURN). A list of commands is displayed at the right edge of the screen. If a number is entered, the value is "pushed" onto the stack. Commands will do various things: e.g., the "+" command will "pop" the top two values from the stack, add their values, and push the result back onto the stack. The "X<>Y" command will swap the top two items on the stack. ">P" will pop the two items on top of the stack, interpreting them as an (x,y) rectangular coordinate pair, transform them to polar coordinates (r, ) and push them both back onto the stack. (">R" performs the inverse operation.) And so on. Examination of the program should remove doubts as to the function of any command.

To add 2 and 3, for example, the following sequence of keys is pressed: 2 <return> 3 <return> + <return>.

All errors (stack underflow, division by zero, control-c interrupt, etc.) are handled by wiping out the stack, printing the word "EPROF", and continuing. Hit reset to end the program.

This program is available at Computerland of Rockville for a \$2.00 copying charge (not including recording media).

Exercises for programmers:

1) Modify the program to emulate your own favorite calculator. Adding, deleting, or changing functions is relatively easy. Try adding an optional hexadecimal display and entry mode. Or, if you dislike RPN, make the program accept an "algebraic" type entry scheme.

2) A weak point in the program is the error-handling routine (lines 1830-1850) where the entire stack is zapped no matter what error was generated. Modify the program to do the "right" things (using your own definition of "right") when different errors occur.

3) Another weak point is in line 260 where it is determined whether a number or a command has been entered. Entering the number 0 as "0.0" will give an error! Incorporate a more sophisticated way of distinguishing numbers from commands into the program.

```

LIST
10 REM -----
20 REM RPN CALCULATOR PROGRAM
30 REM -----
40 SD = 24:SP = 0:M = 0:NC = 24
50 PI = 4 * ATN (1)
60 DIM ST(SD),CM$(NC + 1)
70 ONERR GOTO 1830
80 REM -----
90 REM READ COMMAND TABLE
100 REM -----
110 HOME
120 VTAB 1: HTAB 20: PRINT "COMMANDS:"
130 FOR I = 1 TO NC
140 READ CM$(I)
150 VTAB I: HTAB 30: PRINT CM$(I);
160 NEXT I
170 POKE 33,19
180 REM -----
190 REM COMMAND TABLE
200 REM -----
210 DATA +,-,*,/,^,RCP,CLR,SQR,SIN,COS,TAN,ATN,EXP,LN,STO,RCL,PI,>R,>P,X
    <Y,M+,M-,M*,M/
220 HOME
230 REM -----
240 REM GET NUMBER OR COMMAND
250 REM -----
260 INPUT ":",X$
270 X = VAL (X$)
280 IF X = 0 AND X$ < > "0" THEN 370
290 REM -----
300 REM PUSH NUMBER ON STACK
310 REM -----
320 GOSUB 1710
330 GOTO 260
340 REM -----
350 REM SEARCH COMMAND TABLE
360 REM -----
370 I = 1:CM$(NC + 1) = X$
380 IF X$ < > CM$(I) THEN I = I + 1: GOTO 380
390 IF I = NC + 1 THEN 1830
400 VTAB SP + 1: CALL - 868
410 ON I GOTO 450,520,590,660,730,800,850,890,930,970,1010,1050,1090,1130
    ,1170,1210,1250,1290,1370,1470,1550,1590,1630,1670
420 REM ---
430 REM ADD
440 REM ---
450 GOSUB 1770:P2 = X
460 GOSUB 1770
470 X = X + P2: GOSUB 1710
480 GOTO 260
490 REM -----
500 REM SUBTRACT
510 REM -----
520 GOSUB 1770:P2 = X
530 GOSUB 1770
540 X = X - P2: GOSUB 1710
550 GOTO 260
560 REM -----
570 REM MULTIPLY
580 REM -----
590 GOSUB 1770:P2 = X
600 GOSUB 1770
610 X = X * P2: GOSUB 1710

```

```

620 GOTO 260
630 REM -----
640 REM DIVIDE
650 REM -----
660 GOSUB 1770:P2 = X
670 GOSUB 1770
680 X = X / P2: GOSUB 1710
690 GOTO 260
700 REM -----
710 REM EXPONENTIATE
720 REM -----
730 GOSUB 1770:P2 = X
740 GOSUB 1770
750 X = X ^ P2: GOSUB 1710
760 GOTO 260
770 REM -----
780 REM RECIPROCAL
790 REM -----
800 GOSUB 1770:X = 1 / X: GOSUB 1710
810 GOTO 260
820 REM -----
830 REM CLEAR
840 REM -----
850 SP = 0: GOTO 220
860 REM -----
870 REM SQUARE ROOT
880 REM -----
890 GOSUB 1770:X = SQR (X): GOSUB 1710: GOTO 260
900 REM -----
910 REM SINE
920 REM -----
930 GOSUB 1770:X = SIN (X): GOSUB 1710: GOTO 260
940 REM -----
950 REM COSINE
960 REM -----
970 GOSUB 1770:X = COS (X): GOSUB 1710: GOTO 260
980 REM -----
990 REM TANGENT
1000 REM -----
1010 GOSUB 1770:X = TAN (X): GOSUB 1710: GOTO 260
1020 REM -----
1030 REM ARCTANGENT
1040 REM -----
1050 GOSUB 1770:X = ATN (X): GOSUB 1710: GOTO 260
1060 REM -----
1070 REM EXPONENTIAL
1080 REM -----
1090 GOSUB 1770:X = EXP (X): GOSUB 1710: GOTO 260
1100 REM -----
1110 REM NATURAL LOGARITHM
1120 REM -----
1130 GOSUB 1770:X = LOG (X): GOSUB 1710: GOTO 260
1140 REM -----
1150 REM STORE
1160 REM -----
1170 GOSUB 1770:M = X: GOSUB 1710: GOTO 260
1180 REM -----
1190 REM RECALL
1200 REM -----
1210 X = M: GOSUB 1710: GOTO 260
1220 REM --
1230 REM PI
1240 REM --
1250 X = PI: GOSUB 1710: GOTO 260
1260 REM -----
1270 REM CONVERT TO CARTESIAN

1280 REM -----
1290 GOSUB 1770:R = X
1300 GOSUB 1770:TH = X
1310 X = R * SIN (TH): GOSUB 1710
1320 X = R * COS (TH): GOSUB 1710
1330 GOTO 260
1340 REM -----
1350 REM CONVERT TO POLAR
1360 REM -----
1370 GOSUB 1770:XX = X
1380 GOSUB 1770:YY = X
1390 R = SQR (XX * XX + YY * YY)
1400 IF XX < > 0 THEN TH = ATN (YY / XX)
      + PI * (SGN (XX) - 1) / 2
1410 IF XX = 0 THEN TH = SGN (YY) * PI / 2
1420 X = TH: GOSUB 1710:X = R: GOSUB 1710
1430 GOTO 260
1440 REM -----
1450 REM SWAP TWO TOP NUMBERS
1460 REM -----
1470 GOSUB 1770:P2 = X
1480 GOSUB 1770:P1 = X
1490 X = P2: GOSUB 1710
1500 X = P1: GOSUB 1710
1510 GOTO 260
1520 REM -----
1530 REM ADD TO MEMORY
1540 REM -----
1550 GOSUB 1770:M = M + X: GOSUB 1710: GOTO 260
1560 REM -----
1570 REM SUBTRACT FROM MEMORY
1580 REM -----
1590 GOSUB 1770:M = M - X: GOSUB 1710: GOTO 260
1600 REM -----
1610 REM MULTIPLY MEMORY
1620 REM -----
1630 GOSUB 1770:M = M * X: GOSUB 1710: GOTO 260
1640 REM -----
1650 REM DIVIDE MEMORY
1660 REM -----
1670 GOSUB 1770:M = M / X: GOSUB 1710: GOTO 260
1680 REM -----
1690 REM PUSH NUMBER TO STACK
1700 REM -----
1710 SP = SP + 1:ST(SP) = X
1720 VTAB SP: CALL - 868: PRINT X
1730 RETURN
1740 REM -----
1750 REM POP NUMBER FROM STACK
1760 REM -----
1770 VTAB SP: CALL - 868
1780 X = ST(SP):SP = SP - 1
1790 RETURN
1800 REM -----
1810 REM ALL ERRORS COME HERE
1820 REM -----
1830 HOME : PRINT "ERROR"
1840 FOR I = 1 TO 1000: NEXT I
1850 SP = 0: GOTO 220

```

# Product Review

SOFTSIDE  
PO Box 68  
Milford, NH 03055

For about a year I've seen this magazine advertised in various other magazines and never felt any curiosity toward it. After all, the ads clearly identified it as a TRS-80 magazine. As an APPLE II owner, I felt that it was difficult enough just keeping up with info on the APPLE. Since the magazine was readily available at the recent Philadelphia Computer Conference, and since I had some spare time to kill, I picked up a copy to scan.

My suspicions were confirmed as it was a TRS-80 magazine, apparently concentrating on Basic software for the TRS-80. The issue contained outstanding documentation and a TRS-80 Basic listing for an extensive game. Also it contained write-ups and listings for several smaller programs. Since I liked what I saw, I scanned the back issues. All were in a similar format, containing a well-developed game, smaller programs, and sometimes TRS-80 programming hints or general articles. I found it to be an enjoyable magazine for TRS-80 hobbyists.

So why am I reviewing it for the APPLE Club? It is definitely not for APPLE owners. The software is a different dialect of BASIC; the screen size is different; the graphics are different. Simply, I liked the software I read. The games were high quality and, with the aid of the supporting documentation, I felt that I had enough expertise to convert the programs for APPLE. At the time of this writing, I have been converting for about two weeks. I feel that I have gleaned enough experience to continue converting. I have all the issues of the magazine and will continue to get them, but I most definitely do not recommend this magazine for general or inexperienced APPLE users. I like it and suggest that this is a personal rather than objective enjoyment.

I repeat, so why am I reviewing it for the APPLE Club? It appears that the publisher is to produce the first issue of a new magazine entitled "APPLE SAVED" sometime during January. This is to provide software for the APPLE, as "SOFTSIDE" does for the TRS-80. In fact, there is even no need to rush and subscribe! All registered APPLE II owners (over 20,000 according to the publisher) will receive a sample copy of issue 1 in the mail. If APPLE received your warranty card, you'll receive the magazine. When you get your copy, look it over carefully. I expect that you will like it as much as I expect to like it as well.

Sandy Greenfarb

# PAGE LIST FOR THE APPLE



\*300.388

```

0300- 00 00 A2 04 B5 35 5D 39
0308- 03 D0 06 CA D0 F6 4C 20
0310- 03 A2 04 B5 35 9D 2A 03
0318- 8D 39 03 95 35 CA D0 F3
0320- A9 28 8D 00 03 A9 15 8D
0328- 01 03 60 F0 FD 1B FD A2
0330- 04 BD 2A 03 95 35 CA D0
0338- F8 60 3E 03 1B FD 48 49
0340- 8D F0 05 CE 00 03 D0 0A
0348- A9 28 8D 00 03 CE 01 03
0350- F0 04 68 4C F0 FD A9 15
0358- 8D 01 03 AD 00 C0 10 FB
0360- 48 A9 00 8D 10 C0 68 C9
0368- 83 F0 0B C9 9B D0 E3 A9
0370- 01 8D 01 03 D0 DC A5 33
0378- C9 BE D0 03 4C 03 E0 C9
0380- DD D0 03 4C 00 00 4C 09
0388- FF
    
```

Figure 1.

Remember to type a colon and not a dash, then save the program onto tape. The program can be saved onto tape by typing "300.388W". Once the program is saved onto tape, it may be loaded into the Apple by pushing Reset and then typing "300.388R".

If the system has a disk drive, the program can be saved onto a disk instead of tape. The procedure is as follows:

- Boot up DOS
- Type "POKE 72,0"
- Type "CALL -151" (This gets you into Monitor without killing DOS)
- Type in the program shown in Figure 1
- Use Control C or Control B to get back into BASIC
- Save the program on to disk by typing "BSAVE PAGE LIST, A\$300, L\$89".

The program can be loaded into memory by typing "BLOAD PAGE LIST, A\$300".

Some important points to remember about Page List are:

- When Page List is turned on, DOS is turned off.
- The Page List stays turned on until you turn it off.
- If Page List is left on and you run a program, the program will stop when 20 lines of text have been printed.

## How It Works

Before the Apple prints anything, it first looks at locations \$36 and \$37 to find out where to go next. (The dollar sign indicates that the number is in hexadecimal). On a non-disk system, the computer would go to location \$FDF0 (the output routine in Monitor). When the Apple wishes to input information, it looks at locations \$37 and \$38 to determine where to go next. When the command "CALL 770" is entered, the Initiation Routine (location \$302) first checks to see if Page List is already turned on. If it has not been turned on, the data at locations \$36-\$39 is moved to locations \$32B-\$32E, and the data at locations \$33A-\$33D is moved to locations \$36-\$39. Counter 1 (location \$300) is set to \$28 and Counter 2 (location \$301) is set to \$15.

When Page List is activated, all output is routed first to the Page List routine (location \$33E). There are two counters in Page List that keep track of how much has been printed. Counter 1 counts the number of characters on a line and Counter 2 counts the number of lines printed.

The flow chart, (Figure 2), shows how the counters interact and how the Page List routine works.

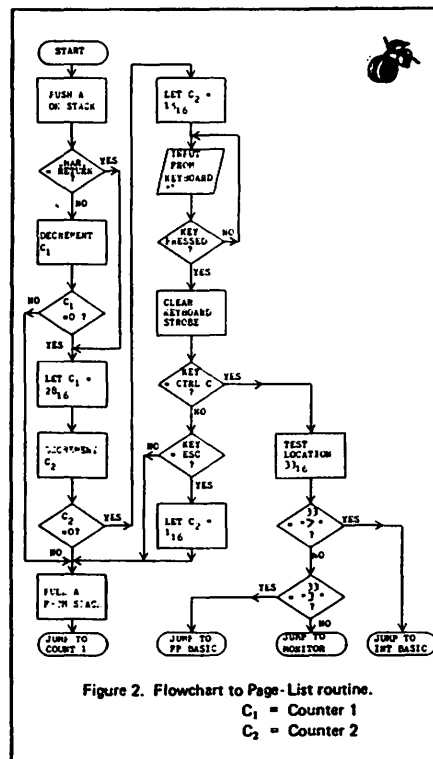


Figure 2. Flowchart to Page-List routine.

C<sub>1</sub> = Counter 1  
C<sub>2</sub> = Counter 2

When "CALL 815" is entered, the Restore routine (location \$32F) moves the data from locations \$32B-\$32E back to locations \$36-\$39.

## Program Relocation

One of the concepts connected with Murphy's Law says, "No matter where you put it, that was the wrong place." While Page List and DOS do not conflict with each other as long as DOS is not rebooted, Page List and the Programmer's Aid #1 HIREs Routines do conflict with each other. I'm sure that there are a lot of other programs out there that conflict with Page List in its present location.

The easiest way to relocate the program is to set all Data Bytes, locations \$300-\$301, \$32B-\$32E, and \$33A-\$33D to \$00 and relocate the program using Apple's Programmer's Aid #1. You will find relocation much easier if you tell the computer that all of the program is in Machine Code and none of it is Data. Once relocated, look at the program listing using Apple's Disassembler. The part you are looking for is the section with four "BRK's" followed by "PHA." The low order byte of the address is placed in the location of the first of the four "BRK's", and the high order byte is placed in the location of the second "BRK". Place \$1B in the location of the third "BRK", and \$FD in the final "BRK". The final step is figuring out the new Basic Calls to turn the program on and off.

If a Programmer's Aid is not available, then program relocation will have to be done by hand. The listing in Figure 3 should be of some help in doing the relocation manually.

300LLLL

```

0300- 00          BRK
0301- 00          BRK
0302- A2 04      LDA  #504
0304- B5 35      LDA  $35,X
0306- 5D 39 03   EOR  $0309,X
0309- D0 06      BNE  $0311
030B- CA        DEX
030C- D0 F0      BNE  $0304
030E- 4C 20 03   JMP  $0320
0311- A2 04      LDA  #504
0313- B5 35      LDA  $35,X
0315- 9D 2A 03   STA  $032A,X
0316- BD 39 03   LDA  $0339,X
031B- 35 35      STA  $35,X
031D- CA        DEX
031E- D0 F3      BNE  $0313
0320- A9 28      LDA  #28
0322- 8D 00 03   STA  $0300
0325- A9 15      LDA  #15
0327- 8D 01 03   STA  $0301
032A- 00        RTS
0323- 00        BRK
032C- 00        BRK
032D- 00        BRK
032E- 00        BRK
    
```

Continued on next page

BY ROBERT D. DIAZ  
2849 W. 235th St. #3  
Torrance, Calif. 90505

The Apple Computer prints on the TV screen at a rate of about 1,000 characters per second. However, most of the Apple users would prefer to read a listing at a slower rate. This article presents a solution to the dilemma: the Page List. The Page List (or listings by page) enables the Apple to list 20 lines of text, stop, and wait for a command from the keyboard to continue or to quit listing.

When the command "CALL 770" is entered into the computer, every twentieth line printed on the TV screen causes the Apple to stop until any key is pressed. The following are the commands used with Page List:

- |                             |  |
|-----------------------------|--|
| CALL 770                    | (Turn on Page List)  |
| LIST                        | (If Page List was turned on, then the computer lists 20 lines and stops) |
| ESC                         | (List one additional line)   |
| CTRL C                      | (Jumps out of the list mode and back into BASIC)                         |
| All other keys except Reset | (Lists 20 more lines)  |
| CALL 815                    | (Turn off Page List)   |

This program is very flexible. It will function with either Integer BASIC, Floating-Point BASIC, with or without DOS, or on any size memory. It also will function with other software such as Microproducts Assembler, with the exception that hitting a Control C will send the program to Monitor Mode rather than returning it to the assembler.

## Loading Page List Into A System


In order to load the Page List into an Apple that does not have a disk drive, first push Reset, then type the program as shown in Figure 1.

032F-	A2 UJ	LDX	\$034
0331-	BD ZA UJ	LDA	\$032A,X
0334-	00 00	STA	\$035,X
0330-	CA	DEK	
0337-	DU F0	BNE	\$0331
0339-	00	RIS	
033A-	00	BRK	
033B-	00	BRK	
033C-	00	BRK	
033D-	00	BRK	
033E-	40	PHA	
033F-	40 0D	EUR	\$000
0341-	F0 00	BEQ	\$0340
0343-	CE 00 UJ	DEC	\$0300
0340-	DJ 0A	BNE	\$0352
0343-	A9 2J	LDA	\$020
0344-	0D 00 UJ	STA	\$0300
034D-	CE 01 UJ	DEC	\$0301
0300-	F0 04	BEQ	\$0350
0352-	00	PLA	
0353-	4C F0 FD	JMP	\$FDF0
0350-	A9 15	LDA	\$015
0350-	0D 01 UJ	STA	\$0301
0358-	AD 00 C0	LDA	\$C000
035E-	10 FB	BFL	\$035B
0360-	48	PHA	
0361-	A9 00	LDA	\$000
0363-	0D 10 C0	STA	\$C010
0360-	00	PLA	
0367-	C9 00	CMP	\$000
0367-	F0 00	BEQ	\$0370
036B-	C9 00	CMP	\$000
036D-	DU E3	BNE	\$0352
036F-	A9 01	LDA	\$001
0371-	0D 01 UJ	STA	\$0301
0374-	DU DC	BNE	\$0302
0370-	A5 00	LDA	\$000
0370-	C9 BE	CMP	\$0BE
037A-	DU 03	BNE	\$037F
037C-	4C 01 EU	JMP	\$E000
037F-	C9 DD	CMP	\$0DD
0301-	DJ 00	BNE	\$0300
0383-	4C 00 UJ	JMP	\$0000
0380-	4C 00 FF	JMP	\$FF00
0389-	FF	???	
038A-	FF	???	
038B-	FF	???	
038C-	FF	???	
038D-	FF	???	
038E-	FF	???	
038F-	FF	???	
0390-	FF	???	
0391-	FF	???	
0392-	FF	???	
0393-	FF	???	

Figure 3.


Closing Comments

I have chosen not to copyright this program in order to provide maximum distribution. It is my hope that most of the Apple owners will be able to utilize this program. Therefore, any person or group wishing to copy this program is more than welcome to do so.



\$322 - STA Counter 1
\$343 - DEC Counter 1
\$30A - STA Counter 1
\$327 - STA Counter 2
\$34D - DEC Counter 2
\$350 - STA Counter 2
\$371 - STA Counter 2
\$315 - STA Tmpdata-1,X
\$331 - LDA Tmpdata-1,X
\$306 - EDB Listloc-1,X
\$318 - LDA Listloc-1,X
\$33A - The low order address for the start of the page list routine.
\$33B - The high order address for the start of the page list routine.
\$30E - JMP Setcounters

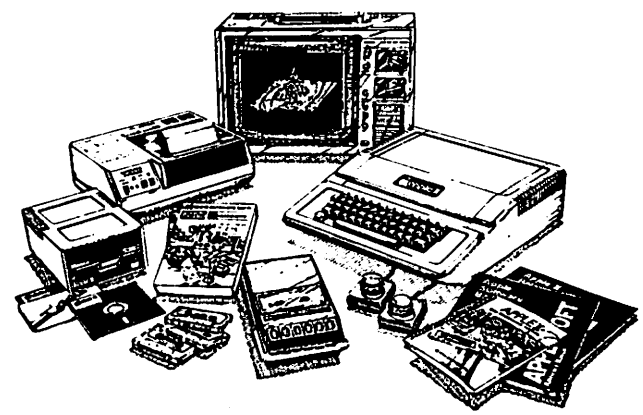
Table 1  
A list of locations that would have to be changed if the program was relocated.



\$33 - Prompt Character
\$300 - Counter 1
\$301 - Counter 2
\$302 - Initiation Routine
\$320 - Setcounters
\$32B - Tmpdata
\$32F - Restore Routine
\$33A - Listloc
\$33C - Low order address of Keyin (\$1D)
\$33D - High order address of Keyin (\$FD)
\$33E - Page List Routine
\$352 - JMP Cout 1
\$37C - JMP Integer Basic
\$383 - JMP FP Basic
\$386 - JMP Monitor

Table 2  
Important Locations

 **apple computer**  
Authorized Dealer



**FREDERICK COMPUTER PRODUCTS, INC.**

MUNICIPAL AIRPORT • FREDERICK, MD. 21701 • (301) 694-8884



## Lowers Paul Rinaldo

One of the reasons why I originally wanted a computer was for text processing. I currently generate correspondence, the *AMRAD Newsletter*, a monthly column for the Washington, DC area amateur radio magazine *AUTO-CALL*, and some articles on an IBM Selectric, not to mention bottles of liquid correction fluid. After purchasing an Apple II computer in late 1978, the desire became very strong to press it into service as a text processor.

There seemed to be very few answers, to begin with, on how to convert the Apple II to handle lower case. One clue was cryptic mention of a "Lower case adapter" (and variously an "upper and lower case adapter") in some ads for just under \$50. A long-distance phone call to Computer Components of Orange County who advertised them revealed that the adapter would not permit natural (like a typewriter) entry of upper and lower case letters from the Apple keyboard. The user must write software to accomplish this. This was somewhat discouraging; the lower case adapter was put off for a while.

Not long after that, I saw a text processor developed by the Muse Company of Baltimore, MD. Later, Muse came up with a disk-based text processor called "Dr. Memory". Recently, they unveiled a newer text processor at the Personal Computing 79 show. While they have some nice features, I passed them over because they use control characters (not the shift key) to change cases and do not display lower case on the video screen.

I also looked at ads and brochures for other text processors and found most of this literature uninformative about the essential features of a text editor for the Apple. Many do not say whether or how they handle lower case. The two choices seem to be to display actual lower-case letters on the screen if the Apple II is equipped with a lower-case adapter or to display INVERSE capital letters instead of lower case. The choices of how you enter a lower case letter are: (a) control characters, (b) hit ESCape once or twice, or (c) use the shift key. Use of the shift key causes problems because shift-P, shift-N and shift-M are normally used for @, ^, and ], respectively. So, if you use the shift for upper and lower case P, N, and M, then you have to hit some other key to tell the computer when you want the @, ^, and ]. Also, this option may or may not call for a slight hardware modification to the Apple keyboard.

All of the above was academic without a printer. In June 1979 I purchased a re-conditioned Diablo printer with keyboard. It became part of the solution and part of the (frustration) problem with lower case. It was understandable why the Apple II keyboard would not produce lower case on the screen because that keyboard was designed for only upper case. It didn't seem reasonable for the Apple II to accept lower case from the Diablo keyboard and, by default, convert it to upper case. A Dan Paymar Lower Case Adapter was purchased about the same time, as was an Applications Unlimited so-called "Word Processor". I used "so-called" here to signify that it really isn't a "word-oriented" or "cursor-oriented" text processor at all -- it is a "line-oriented" processor. That means that the way you type text in, line by line, is the way it will be printed. It is not nearly as flexible as the word-oriented processor which let you type in an endless stream of words and then enable you to print it out in any line length you wish. The line-oriented processor has its place, no doubt, but it was clear that I needed a word-oriented processor. Another thing I did not like about the Applications Unlimited processor was the need to use one or two ESCapes to change case. The need for doing something like this was understandable when the Apple II keyboard was used because it has its limitations. However, it was even more frustrating to have to do it with a full ASCII keyboard such as the one supplied with the Diablo. After making some minor changes, I was able to get the Apple to display lower case when the Diablo keyboard keyed in lower case. After that small victory, it was a shock to see the Apple display the lower case characters as INVERSE upper case while outputting the stored text to the Diablo which was printing lower case. Finally, the documentation supplied with the word processor was inadequate in many respects. I probably wouldn't have noticed the documentation shortcomings except that I had just received an Apple Bulletin Board System (ABBS)

package from the same outfit... it was well documented.

Speaking of the ABBS, one day I was scanning the messages in the AMRAD message system (703) 281-2125 and magically a few lines of lower case came on the TV screen. Eureka! This happened with just the Apple, the lower case adapter, and the D.C. Hayes Micromodem -- no special software of any kind. About that time, someone called me to ask if I could use my printer to print his text that he could send me over the modem. That seemed simple, so I activated the printer port with a PR#1, etc. Much pain. The Apple High-Speed Serial Interface board used to drive the printer, when activated, converted the lower case letters to upper case not only on the printer but on the TV screen as well. I still haven't figured that one out.

That's about where things stand at the moment. I have a text processor that I don't like. I'm not really interested in buying another, sight unseen. At least I think that I have a clearer idea of how a good word-oriented text processor should work with the Apple. Maybe it sounds a little drastic but I'm seriously thinking of replacing the Apple keyboard with a Cherry Pro keyboard which has a full ASCII character set. The Pro will physically fit but it draws about 350 mA or about 250 mA more than the Apple will supply through the keyboard connector.

### References:

1. David Minch, "Dan Paymar's Lower Case Adapter - A Review," rainbow Newsletter, July 1979.
2. David Wilkerson, "Lower-casing it on the Apple II," ABACUS, April 1979.
3. Larry Danielson, "Lower Case Mod," ABACUS, April 1979.
4. Documentation supplied with Dan Paymar Lower Case Adapter, Dan Paymar, PO Box A-133, CS 6800, Costa Mesa, CA 92627.
5. Muse News, Numbers 1 and 2, the Muse Co., 7112 Darlington Drive, Baltimore, MD 21234.
6. Letter describing "BIG-EDIT" text processor the the Apple II, Garvey, Martin and Sampson, Inc., 723 Dover Ave., Middletown, OH 45042.
7. Data sheet on EDIT, Apple II Disk Text Editor for the Apple II, Services Unique, Inc., 2441 Rolling View Dr., Dayton, OH 45431.
8. Data sheet on The Word Weaver Text Processing System, Versions 1.0 and 2.1, Huelsonk Products, 15703 Midvale No., Seattle, WA 98133.

- Delicious Apples. NO SIZE WASH STATE RED LB 69¢
- Delicious Apples. 72 SIZE BAKING LB 69¢
- Rome Apples... LB 59¢



Whimsy Department

ADDING COLORS TO APPLE-II HI-RES  
(nullifies warrantee)

1. Remove the APPLE-II PC board from its enclosure

(a) Remove the ten (10) screws securing the plastic top piece to the metal bottom plate. Six (6) of these are flat-head screws around the perimeter of the bottom plate and four (4) are round-head screws located at the front lip of the computer. All are removed with a phillips head screwdriver. Do not remove the screws securing the power supply or nylon insulating standoffs.

(b) Lift the plastic top piece from the bottom plate while taking care not to damage the ribbon cable connecting the keyboard to the PC board. This cable will have to be disconnected from one or the other.

(c) Disconnect the power supply from the PC board.

(d) Remove the #8 nut and lockwasher securing the center of the PC board. These will not be found on the earlier APPLE-II computers.

(e) Carefully disengage each of 6 nylon insulating standoffs from the PC board. (7 on earlier versions).

(f) Lift the PC board from the bottom plate.

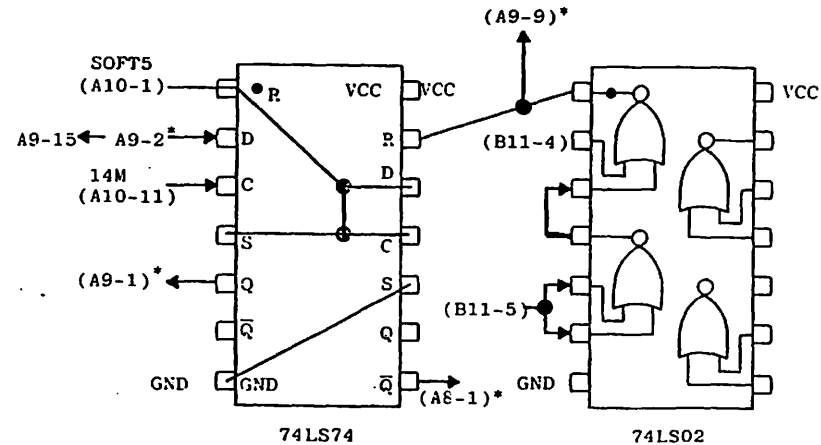
2. Above the board wiring method

(a) Lift the following IC pins from their sockets.

- A8-1
- A8-6
- A8-13
- A9-1
- A9-2
- A9-9

(b) Mount a 74LS74 (dual C-D flip-flop) and a 74LS02 (quad NOR gate) in the APPLE-II breadboard area (A11 to A14 region).

(c) Wire the following circuit (\* indicates that wiring is to a pin which is out of its socket).



- (B8-14) → (A8-C)\*
- (B8-7) → (A8-13)\*

3. Below the board wiring method (for neater appearance)

(a) Desolder all pins of socket A8. Lift the socket and its 74LS257 IC off the PC board taking care not to destroy it. Cut the trace between pins 6 and 13 of A8 on the top side of the board. Also cut the trace between pins 13 and 15 on the top. Reinsert socket A8 and the 74LS257. Be careful!

(b) Cut traces going to the following IC pins on the bottom of the APPLE-II board. Each pin should have a single trace going to it. Be careful!

A8-1	A9-1
A8-6	A9-2
A8-13	A9-9

(c) Connect A8-15 to ground (A7-8 on the keyboard socket is a nearby ground).

(d) Mount the 74LS74 and 74LS02 as per step (b) of the 'above the board' wiring method.

(e) Wire the circuit of the 'above the board' wiring method, step (c). All wires are on the bottom of the APPLE-II board and no IC pins need be removed from their sockets or soldered to.

4. Reassemble the APPLE-II and make sure it is operational. If not, check all wiring very carefully. Make sure that all chips are in their sockets and properly oriented.

5. The following color values are now applicable to the HI-RES subroutines.

BLACK2	128
ORANGE	170
BLUE	213
WHITE2	255

For example, the program below draws an orange line from (10,20) to (200,140). It is assumed that the HI-RES routines are already in memory locations \$800-\$BFF.

```
0  XO = YO = COLR
5  INIT = 2048 : PLOT = 2830 : LINE = 2836
7  ORANGE = 170 : CALL INIT
10 XO = 10 : YO = 20 : COLR = ORANGE : CALL PLOT
20 XO = 200 : YO = 140 : CALL LINE
30 END
```

# APPLE USER GROUPS

A list follows of all Apple User Groups that we could come up with at press time. This list has been compiled from many sources and, undoubtedly, is incomplete and probably has some errors too - and definitely a bit of duplication. Nevertheless, and for what it is worth, here it is..... We would appreciate being notified of any changes. Phone numbers are found to the right of the names.

mc

G. K. INMAN APPLE-HOLICS  SRA BOX 1313 ANCHORAGE, AK 99502	9073441300	LARRY GOGA SAN FERNANDO VALLEY 6502 USERS CLUB 3816 ALBRIGHT AVENUE LOS ANGELES, CA 90066	2133986086
J T ROGERS PRINTOUT/APPLEPEEL  1161 SHADES CREST ROAD BIRMINGHAM, AL 35226	0	DAVID SMITH NORTH ORANGE COUNTY COMPUTER CLUB 607 NORTH TWILIGHT PLACENTIA, CA 92670	7149939939
TERRY WOODWARD APPLE CORPS COMPUTER CENTER, INC 433 VALLEY AVENUE PLAZA BIRMINGHAM, AL 35209	2059428567	- APPLE USER GROUP VIDEO GAMES & COMPUTERS 301 BALBOA SAN FRANCISCO, CA 94118	4152218500
D JOHNSON DATA BITS DATA COPE 5706A W 12TH ST LITTLE ROCK, AR 72204	5016668588	. . . . APPLE USERS GROUP CARR ELECTRONICS 5811 GEARY BLVD SAN FRANCISCO, CA 94121	4156684243
ED AVELAR ABACUS  2950 JENNIFER DRIVE CASTRO VALLEY, CA 94546	4155382431	KEN SILVERMAN APPLE CORE LIBRARY EXCHANGE PO BOX 4816 SAN FRANCISCO, CA 94101	4158785382
KIP REINER PRES, ORIGINAL APPLE CORPS APPLESAUCE 12904 MAGNOLIA CHINO, CA 91710	2139021220	MELVIN WONG  APPLE-BIZ 301 BALBOA SAN FRANCISCO, CA 94118	4152218500
SCOTT STARKWEATHER APPLE USERS GROUP  11074 SAN PABLO AVENUE EL CERRITO, CA 94530	4152335010	JIM HOYT SILICON APPLE PROGRAMMING SOCIETY 2485 ROSSOTTO DRIVE SAN JOSE, CA 95130	4083743680
--- APPLE CORE AVIDO ELECTRONICS 2210 BELLFLOWER ROAD LONG BEACH, CA 90815	2135980444	. . . . BYTE SHOP 4 WEST MISSION STREET SANTA BARBARA, CA 93101	8059662638
. . . . APPLE CORPS AVIDO ELECTRONICS 2210 BELLFLOWER ROAD LONG BEACH, CA 90815	2135980444	LOY SPURLOCK APPLE BYTE USERS GROUP  14052 E FIRESTONE BLVD SANTA FE SPRINGS, CA 90670	2139212111
SARKIS KOUZOUJIAN  COMPUTERLAND OF LOS ALTOS 4546 EL CAMINO REAL LOS ALTOS, CA 94022	4159418154	--- THE APPLE PICKERS SANTA ROSA COMPUTER CENTER 604 SEVENTH STREET SANTA ROSA, CA 95404	0
-- L.A. APPLE USERS GROUP  11911 WILSHIRE BLVD LOS ANGELES, CA 90025	2138268499	MARK WOZNIAK RECREATIONAL COMPUTER CENTER COMPUTERS PLUS 1324 S. MARY SUNNYVALE, CA 94087	4087357480



DAVID KAY APPLE CORE COMPUTERLAND OF THOUSAND OAKS 171 E THOUSAND OAKS BLVD #104 THOUSAND OAKS, CA 91320	0	FRESTON LOVE APPLE USER GROUP DATAMART INC 3001 N FULTON DRIVE ATLANTA, GA 30305	4042660336
..... APPLE USERS GROUP COMPUTERLAND 1315 YGNACIO VALLEY ROAD WALNUT CREEK, CA 94596	0	GARY LITTLE APPLE B.C. COMPUTER SOCIETY  101-2044 W THIRD AVENUE B.C. CANADA, 0	6047317886
TERRY TAYLOR APPLE PI/THE SEED  12319 E BATES CIRCLE AURORA, CO 80014	0	R ETTORE EUROPE COMMUNITY COMPUTER CLUB 59, RODE BEUKENLAAN 1970 WEZEMBEEK-OPPEM BELGIUM, 0	0
CARL GRIMES DENVER AMATEUR COMPUTER SOCIETY 1380 S. SANTA FE DENVER, CO 80223	3037598969	RALPH DAWSON APPLE BRITISH COLUMBIA 2922 EAST 25TH AVENUE VANCOUVER, B.C CANADA, 0	0
AUSTIN R BROWN JR APPLE PI  407 PEERY PARKWAY GOLDEN, CO 80401	3032795388	SCHRAEN DOMINIQUE APPLE OEDIP 8, PLACE STE-OPPORTUNE-75001 PARIS FRANCE, 0	-5084621
GLEN BRENNAN APPLEFIELD FARE COMPUTERLAND OF FAIRFIELD 2475 BLACKBROOK TURNPIKE FAIRFIELD, CT 6430	2083742227	ALFREDO BUZALI MICROCOMPUTER CLUB FTE DE QUIJOTE #5  MEXICO 10, D.F., 0	-5892279
MARK GOLDFARB APPLELIST  55 PARDEE PLACE NEW HAVEN, CT 6515	2035624907	---- APPLE USER GROUP EUROPE POSTFACH 4068 D-4320 HATTINGEN WEST GERMANY, 0	0
BILL STEWART WASHINGTON AMATEUR COMPUTER SOCIETY 4201 MASS AVE, N.W. #168 WASHINGTON, DC 20016	2027220210	BILL MARK HONOLULU APPLE USERS SOCIETY  98-1451-A KAAHUMANU ST AIEA, HI 96701	8084882026
JAMES H HIGGINS COMPUTERLAND OF NEWARK ASTRO SHOPPING CENTER KIRKWOOD HIGHWAY NEWARK, DE 19711	3027389656	DENNIS NYHAGEN APPLE USER GROUP  7110 C OHANA-NUI CIRCLE HONOLULU, HI 96818	0
PAT FIORENTINO APPLE USER GROUP  2201 PONCE DE LEON BLVD CORAL GABLES, FL 33134	0	EARL KEYSER  22 CLOVER LAND MASON CITY, IA 50428	0
- - - A.C.E.S.  671 N.E. 56TH STREET FT. LAUDERDALE, FL 33334	3057724768	LARRY BUGBEE  2874 ITHACA BOISE, ID 83705	2083629132
DAVID HALL MIAMI APPLE USERS GROUP  2300 NW 135TH STREET MIAMI, FL 33167	0	MIKE URSO MICROPROCESSOR USER GROUP  641 WOODLAWN AURORA, IL 60506	0
VICTOR STEEB APPLE USER GROUP SOUTHERN MICROCOMPUTER CO 5901E NORTHWEST 151ST STREET MIAMI LAKES, FL 33169	3058217401	KEN ROSE NORTHWEST SUBURBAN APPLE II USERS GROUP 650 POMPANO LANE PALATINE, IL 60067	3123596723

JOHN GREVE QUAD CITY COMPUTER CLUB  4211 7 AVENUE ROCK ISLAND, IL 61201	3097868197	... .. THE MICHIGAN APPLE  32905 W 12 MI ROAD STE 320 FARMINGTON HILLS, MI 48018	3139795298
-- -- CHICAGO AREA COMPUTER HOBBYISTS EXCHANGE (CACHE) PO BOX 52 SOUTH HOLLAND, IL 60473	3128491132	-- -- MICHIGAN/APPLE-GRAM  PO BOX 551 MADISON HEIGHTS, MI 48071	0
JERRY FEIL APPLE PIE  17318 S LOCUST TINLEY PARK, IL 60477	3125328244	DAN BUCHLER  MINI'APP'LES 13516 GRAND AVENUE SOUTH BURNSVILLE, MN 55337	6128905051
JOE TORZEWSKI APPLE LIBRARY  51625 CHESTNUT ROAD GRANGER, IN 46530	0	.. . . . FUTUREWORLD, INC. MICRO & PERSONAL COMPUTER CL 12304 MANCHESTER ROAD ST LOUIS, MO 63131	3149654540
DOUG MCINTOSH INDY APPLE PICKERS HOME COMPUTER CENTER 2115 E 62 STREET INDIANAPOLIS, IN 46220	0	CREIGHTON CALFEE THE APPLE JACKS OF ST LOUIS  PO BOX 8452 ST LOUIS, MO 63132	0
.. . . . APPLEBUTTER - 10049 SANTA FE DRIVE OVERLAND PARK, KS 66212	0	NOEL MOSS ST LOUIS AREA COMPUTER CLUB  PO BOX 28924 ST LOUIS, MO 63132	3148624040
ALLEN SIMPSON THE B.R.A.N.C.H.  4661 TUPELO STREET BATON ROUGE, LA 70808	5049240636	ODEL SMALL APPLE JACKS  PO BOX 24202 ST. LOUIS, MO 63130	0
.. . . . SOUTHEASTERN SOFTWARE 7270 CULPEPPER DRIVE NEW ORLEANS, LA 70126	0	ALEX POPPER APPLE CORE COMPUTER CLUB  3915 E INDEPENDENCE BLVD CHARLOTTE, NC 28205	7045235107
DAVE MITTON NEW ENGLAND COMPUTER SOCIETY  PO BOX 198 BEDFORD, MA 1730	6174933154	NANCY TENELL GREEN APPLES GREENSBORO BYTE SHOP 218 NORTH ELM STREET GREENSBORO, NC 27410	9192752983
----- NEATNOTES NEW ENGLAND APPLE TREE 872 MASSACHUSETTS AVENUE CAMBRIDGE, MA 2139	0	WILBUR ANDREWS CAROLINA APPLE CORE  5212 INGLEWOOD LANE RALEIGH, NC 27609	9197873509
DONALD M ISAAC APPLESEED  17 SAXON ROAD WORCESTER, MA 1602	0	RUSS GENZMER APPLE SAUCE OF LINCOLN-OMAHA C/O TEAM ELECTRONICS 2055 'O' STREET LINCOLN, NE 68510	4024354467
JOE LOUIS  MARYLAND APPLE CORPS 13A ALLEGHENY AVENUE TOWSON, MD 21264	3012960520	DICK SEDGEWICK  100 HORNE STREET DOVER, NH 3820	0
KEVIN PARKS MARYLAND APPLE II USERS GROUP COMPUTERLAND UNLIMITED, INC. 907 YORK ROAD TOWSON, MD 21204	3013211553	DAN FISCHLER  COMPUTER LAB OF NEW JERSEY 141 ROUTE 46 BUDD LAKE, NJ 7828	2016911984

STEVE TOTH APPLE TREE OF CENTRAL N.J.  1411 GREENWOOD DRIVE PISCATAWAY, NJ 3854	2019687498	WILL NEWMAN II APPLE PORTLAND PROGRAM LIBRARY EXCHANGE 1915 N.E. COUCH PORTLAND, OR 97232	5032335711
- - - - COMPUTER ENCOUNTER APPLE USER GROUP 2 NASSAU STREET PRINCETON, NJ 9540	6092948757	KEN HOGGATT APPLE PORTLAND PROGRAM LIBRARY EXCHANGE 9195 S.W. ELROSE COURT TIGARD, OR 97223	5036395505
EARL J NIELSEN THE APPLE CORPS PERSONALIZED COMPUTER SERVICES 1803 CORTE DEL RANCIERO ALAMOGORDO, NM 88310	5054378447	NEIL LIPSON PHILADELPHIA AREA COMPUTER SOCIETY 29 S. NEW ARDMORE AVENUE BROOMALL, PA 19008	2153566183
APPLE BYTER'S CORPS (MEETING PLACE) BUFFALO SAVINGS BANK 3990 SHERIDAN DRIVE ANHERST, NY 14226	7168475800	- - - APPLE USER GROUP COMPUTERLAND OF HARRISBURG 4644 CARLISLE PIKE MECHANICSBURG, PA 17055	0
CHARLES KOLLETT  32 N BREWSTER LANE BELLPORT, LONG ISLAND, NY 11713	5162860198	DICK MOBERG PHILADELPHIA AREA COMPUTER SOCIETY (PACS) PO BOX 1954 PHILADELPHIA, PA 19105	2159233299
- - - NYC USERS GROUP THE DRYSDALE SECURITY 55 WATER STREET NEW YORK, NY 10004	5165794295	- - - APPLE USER GROUP THE COMPUTER HOUSE 1000 GREENTREE ROAD PITTSBURGH, PA 15220	4129211333
NEIL SHAPIRO NYC USERS GROUP COMPUTER/HART OF NEW YORK 119 MADISON AVENUE NEW YORK, NY 10016	2126867923	RICHARD C SECRIST APPLE PI  RT #12, CHEROKEE HILLS SEVIERVILLE, TN 37862	0
- - - ASD OFFICE SYSTEMS THE MID-HUDSON APPLE CORP RT 55 - VANWYCK PLAZA POUGHKEEPSIE, NY 12603	0	- APPLE CORPS COMPUTERLAND OF AUSTIN 3300 ANDERSON LAND AUSTIN, TX 78757	4124525701
GARY WEIR APPLE BYTER'S CORPS  225 WALTON DRIVE SNYDER, NY 14226	7168393486	BOBBIE FERRELL THE APPLE CORPS - GREENHILL SCHOOL, FULTON BLDG 14255 MIDWAY ROAD DALLAS, TX 75240	2146611211
. . . . . COMPUTER SOLUTIONS DAYTON AREA APPLE USERS 1932 BROWN STREET DAYTON, OH 45409	0	R. V. COLLINS APPLE BARREL  12502 BEXLEY HOUSTON, TX 77099	0
DAN WATSON DAYTON MUSEUM OF NATURAL HISTORY 2629 RIDGE AVENUE DAYTON, OH 45414	5132732348	BILL HYDE APPLESEED THE COMPUTER SHOP 6812 SAN PEDRO SAN ANTONIO, TX 78216	5128280553
JOHN ANDERSON APPLE-SIDER  5707 CHESAPEAKE WAY FAIRFIELD, OH 45014	5138291340	PHILIP W JACKSON  COMPUTER SOLUTIONS SUITE 124A 9200 BROADWAY SAN ANTONIO, TX 78217	8002927452
JERRY HENSHAW  THE TULSA COMPUTER SOCIETY PO BOX 1133 TULSA, OK 74101	9188367364	BRUCE LERNER APPLE TRIPOWER ELECTRIC 90 E 4500 STREET MURRAY, UT 84107	9012620860

# Bytes from the APPLE Pi

Sandy Greenfarb

This column is intended to be a forum for presenting ideas, discoveries, techniques and such that might be of benefit to other APPLE users and do not warrant a full article. Length does not denote importance and sometimes significant gems go unsaid because their authors felt they were too short. I am going to start this off with a couple of items of my own and hope that others will follow suit and contribute their short tidbits. How often this column appears depends on the support of the membership.

Recording APPLE Music: While rereading the red APPLE II Reference Manual (137th time), I discovered a reference to \$C02X, toggle cassette output. As an experiment, I took a music demo and changed the references in the program from \$C030 (toggle speaker output) to \$C020. The result was music that could be recorded and/or played out loud on my cassette recorder.

APPLESOFT - Symbolic subscript in DIM statement - In order to keep my daughter interested in HANGMAN, I constantly modify the list of words in my DATA statements. I found that a symbolic subscript simplified my program changes. Hopefully, the following technique will be of benefit to others:

```
10 NUMBER=6: REM NUMBER OF WORDS IN DATA STATEMENTS
20 DIM WORD$(NUMBER)
30 FOR I=1 TO NUMBER: READ WORD(NUMBER): NEXT
..
70 CURRENT$=WORD$(1+INT(NUMBER*RND(1))): REM SELECT WORD
..
199 DATA WORD1, WORD2, WORD3, WORD4, WORD5, WORD6
```

Now whenever I change the number of words, line 10 is the only other change required. Note that NUMBER must be defined before the DIM statement.

>32767 (48K machine): Some Integer Basic programs, for one reason or another, save pointers in a variable for use in the program, i. e., HIGHMEM=PEEK(77)\*256+PEEK(76). These same programs failed with a >32767 ERROR when I moved up to 48K. The following line substitution cured this problem. HIGHMEM=PEEK(77)MOD128\*256+PEEK(76)-16384\*(PEEK(77)>127)-16384\*(PEEK(77)>127)

TINY PASCAL - Combining libraries from disk: Users of Tiny Pascal quickly discover the need for maintaining libraries of procedures and functions for use with their programs. To use one source library, simply load the library and write the program around it. I discovered the following method to append one library to another. In Pascal monitor mode, load the first library. Start the Editor and this should give you a statistical message of bytes used for your source (for this example I'll use 5000-5CFE). Next return back to Monitor and execute a DOS command loading the second library and the last address used by the first library (BLOAD LIB2, A\$5CFE). The libraries are now appended. The reason this works is that Tiny Pascal uses a \$FF as the last character of a source file to indicate End-of-File. By overwriting the first End-of-File mark, you have connected the files.

## Parallel Interface

Dave Skillman

I needed a large number of parallel lines for a data link so I bought an Apple prototyping card and added an 8255 chip which has 24 I/O pins that can be programmed as either input, output, or mixed combinations. The advice that comes with the Apple card is quite intimidating and states outright that an unbuffered I/O chip probably won't work. Mine seems to work just fine, but I haven't pushed it very hard. I offer it to those who might like to tinker. I used two 15196s and one 1504, to allow for further addition to the board, but I think that two transistors and the resistors are all that's needed to make the 8255 work. The cost was \$90 (card), \$7 (8255), \$3 (other chips and sockets) and \$3 (25-pin connector to the external world) for a total of \$43. The bulk of it was the card, but there's plenty of room left over for UART's, clocks, etc.

This month's BYTE has a long article on the 8255. That implementation is far more complex due to the S-100 bus restrictions. Referring to the 8255 pin diagram on page 125 of that issue, the following is a list of connections.

8255 pin	function	Apple I/O connector pin
9	address, bit zero	2
8	address, bit one	3
6	chip select low	1
7	ground	26
26	+5 volts	25
27 to 34	data, bit 7 to 0	42 to 49
36	write when low	18
5	read when low	19
35	reset when high	31 (inverted)

The inversion of a signal can be accomplished with a resistor and a transistor, or with a TTL 74 1504 chip. If anyone is interested in more details, feel free to contact me. (301) 474-0653



# Adjusting the DISK II SPEED

Mark L. Crosby

First, let me explain that I do not recommend you immediately tear your disk apart to adjust its speed as that will certainly nullify your warranty. However, if you are somewhat mechanical and your warranty has expired, then this is for you.

You do not need any electronics experience nor any special tools other than either a (a) neon power tester or (b) a fluorescent light, and a tiny screwdriver.

First, turn off the power to your Apple and remove the Disk Unit from its Controller Card (leave the card in the Apple). Now that the Disk unit is safely disconnected from any power source it is safe to open it. Remove the 4 screws from the bottom of the unit which hold the cover to the base-plate. Slide the cover backwards - away from the disk gate - until it is free. This may require some considerable and careful effort as the cover fits tightly against the base-plate and associated structural supports inside the unit. Just be sure the multi-wire cable is not caught on the cover as you slide it backwards. After the cover has been removed, turn the disk unit over and place the cover aside for a moment.

Now remove the remaining 4 screws on the bottom of the unit. This will separate the base-plate from the remainder of the unit. Before you try to remove the base-plate (after the screws are out), unclip the cable clamp holding the multiwire cable to the back of the base-plate and slide the cable out of the clamp. Now lay the base-plate aside and turn the disk unit upside down.

Now reconnect the multiwire cable to the Controller Card and power up the Apple II. Attempt to boot the DOS (do not place a floppy disk in the Disk unit) by hitting the Ctrl-P return sequence. This will start the disk motor turning.

On the very center part of the unit is a strobe disk calibrated for both 50 and 60 Hz. So bring over the neon light or fluorescent lamp, turn it on and watch the pretty lines moving just like your record turntable. Note which lines are for 60 Hz!!!

Now the "hard" part. You do have a tiny screwdriver don't you? Well, on the back apron of the disk unit is a small printed circuit board and peeking up at you from one end on the upper edge (which is really the bottom edge but the unit is upside down now) is a small 10-turn potentiometer. See figure. Actually, this may need only a slight trimming up so a small turn may produce significant results. As you turn the pot, watch the strobe lines until they come to a halt - or as nearly non-moving as possible. You are finished.

Reverse the procedure to put everything back together - remember to turn off the power first, remove the cable from the controller card, clip the cable back onto the base-plate, put the 4 screws back, put the cover back by sliding it from the rear forward and, finally, put the remaining 4 screws back.

Simple? Definitely.

p.s. This is how the "pros" do it.

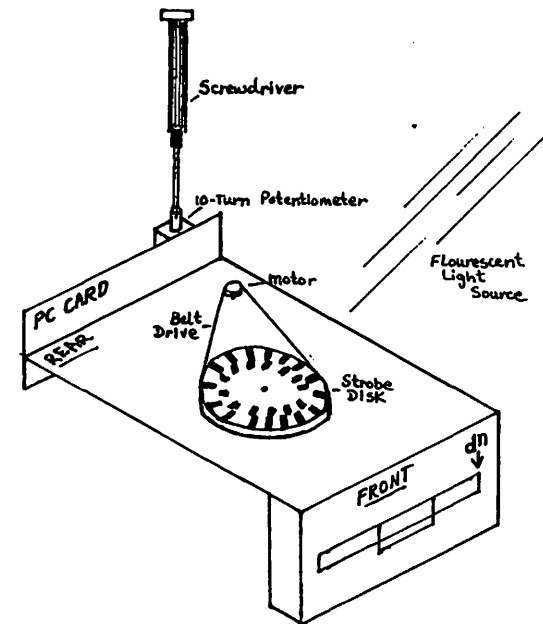
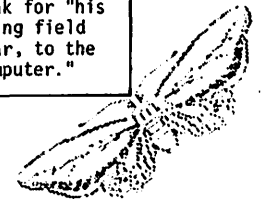


fig. 1

Stephen Wozniak, Vice President of research and development for the Apple Computer Company, Cupertino, California, received the Association for Computing Machinery (ACM) Grace Murray Hopper award at the ACM annual conference last month in Detroit. The Hopper award cites Wozniak for "his many contributions to the rapidly growing field of personal computing and, in particular, to the hardware and software for the Apple Computer."



Season's Greetings

## COMPUTER EMPORIUM

featuring:

 apple computer inc.

 commodore

exidy

cromemco

1990 K St.,NW  
D.C.

466-3367

1984 Chain Br. Rd.  
VA.

821-8334

service shop:  
821-8394



call for  
new xmas hours!

## What's Where in the APPLE

Professor William F. Luebbert  
Dartmouth College  
Hanover, NH 03755

Whether you are programming in BASIC or assembly language, a memory map helps save time, reduce program size and improve performance. This is the most complete and up to date APPLE memory map ever published.

To get the most out of an APPLE, or any other computer with limited resources, it is important to know a good deal about the hardware and software environment.

When one graduates from simple programs to more ambitious programs involving careful control of man-machine interaction, analog to digital or digital to analog conversion, extensive use of computer graphics, the control of external devices, database management, sorting, word-processing or any of a wide variety of interesting tasks, this knowledge tends to become more important. When (and if) one gets into real time programming, adding his own specialized interfaces, performs activities where one must get the absolute maximum speed or gets into other situations where machine language programming is appropriate, it becomes critical.

Not every serious programmer needs to become a machine language level programmer. However, good programmers know that when the computer is running their programs there is a good deal of machine language code in the machine providing an operating environment for their programs. This operating environment typically includes the system monitor, a BASIC interpreter and possibly a disk operating system (DOS) and/or extra ROM packages.

When one looks at interesting programs described in magazines and user group newsletters, he finds that these programs often contain PEEKs, POKEs and CALLs. These are commands which are extensions of BASIC (or other higher

level languages). They are provided to allow one to interface with the computer hardware, operating environment software, and other machine language programs or subprograms.

PEEKs, POKEs and CALLs all refer to memory locations which are identifiable as to what they contain or what they do. a PEEK examines the contents of a specified memory location and allows one to use that content in a program. POKE changes the contents of a designated memory location to some specified value. It can be used to change parameters of the operating environment or to set up or change pieces of program or data. A CALL transfers program control to a particular memory location and sets up a return linkage for transfer back to the CALLing routine in the user's program.

Pieces of the monitor or some other parts of the operating environment can often be accessed via CALLs, POKEs and PEEKs to modify system operation or to perform desired functions without the necessity of additional code. Usually this code has been carefully written in machine language and optimized by good programmers, so it runs faster and takes less space or less computer time than the same function would require if programmed totally by the user.

A programming manual intended for serious programmers should supply some sort of memory map and information about the most important and frequently used PEEKs, POKEs and CALLs. A good memory map can show the user where he can get information from the

computer, what potentially useful software is available but perhaps hidden away inside the computer, and the "hooks" provided to perform a wide variety of functions by means of CALLs, POKEs and/or PEEKs. Often it becomes the most well-worn section of the manual. Once programmers begin using it as a source of information, they begin to wish for a more complete atlas which will let them find more and more information and guide them in their own explorations inside the computer and its software.

The memory map presented here was developed initially as a programming aid for my own personal programming. Important sources of information for its creation included the *APPLESOFT II Manual*, the *APPLE Reference Manual*, *WOZPAC* and various issues of *MICRO*, *Call-Apple* and *NEAT* as well as my own investigations inside the computer.

The map is being circulated for comment, correction and modification by many of the more active members of the New England Apple Tree User's Group. They have suggested valuable changes, corrections and additions. Inevitably there will still be errors and omissions. For these I beg your indulgence.

This memory atlas is stored on-line on the Dartmouth Timeshare System in a database which can be used for selective retrieval and report generation using standard database management software. The author would appreciate corrections or suggested changes or additions. Please mail them to him at Hinman, Box 5166, Dartmouth College, Hanover, NH 03755.

LUEBBERT'S COPYRIGHT APPLE MEMORY ATLAS - MICRO MAGAZINE VERSION

HEXLOC	DECLOC	NAME	USE
0000-000F	0-255		HARDWARE PAGE ZERO
0000-0005	0-5		JUMP INSTRUCTIONS TO CONTINUE IN APPLESOFT
0000-0001	0-1	ROL*ROH	SHEET-16 (16-DIT INTERPRETER) REGISTER RO
0000	0	LOCO	MONITOR MEMORY LOCATION 'LOCO'
0001	1	LOCI	MONITOR MEMORY LOCATION 'LOCI'
000A-000C	10-12		LOCH FOR USER FUNCTION'S JUMP INSTRUCTION
000D-0017	13-23		GENERAL PURPOSE COUNTERS/FLAGS FOR APPLESOFT
001A-001B	26-27		HI-RES GRAPHICS ON-THE-FLY SHAPE POINTER
001A-001B	26-27	SHAPEL*SHAPEH	HI-RES POINTER TO SHAPE LIST
001C	28		HI-RES GRAPHICS ON-THE-FLY COLOR BYTE
001C	28	HCOLORI	HI-RES RUNNING COLOR MASK
001D	29		HI-RES GRAPHICS HIGH-ORDER BYTE OF STEP COUNT FOR LHM
001E-001F	30-31	R1L*R1SH	SHEET-16 (16-DIT INTERPRETER) REGISTER R1S
0020-004F	32-79		APPLE II SYSTEM MONITOR RESERVED LOCATIONS
0020	32	WMDLEFT	SCROLLING WINDOW LEFT SIDE (0-39 OR 40-62)
0021	33	WMDWTH	SCROLLING WINDOW WIDTH (1-40 OR 41-62)((WMDLEFT+WMDWTH)*40)
0022	34	WMDTOP	SCROLLING WINDOW TOP LINE (0-23 OR 24-46)
0023	35	WMDBTM	SCROLLING WINDOW BOTTOM LINE (0-23 OR 24-46)((WMDBTM-WMDTOP)*40)
0024	36	CH	CURSOR HORIZONTAL POSITION (0-79 OR 80-97)
0025	37	CV	CURSOR VERTICAL POSITION (0-23 OR 24-46)
0026-0027	38-39	DBASL*DBASH	LOW-RES GRAPHICS POINTER TO LEFTMOST BYTE OF CUR PLOT LINE
0026-0027	38-39	HDBASL*HDBASH	HI-RES GRAPHICS ON-THE-FLY BASE ADDRESS
0028-0029	40-41	DASL*DBASH	MONITOR BASE ADDRESS POINTER
002A-002B	42-43	BAS2L*BAS2H	MONITOR BASE ADDRESS POINTER 2
002C	44	H2	LOW-RES COLOR GRAPHICS #2
002C	44	LPH2M	MONITOR MEMORY LOCATION 'LPH2M'
002C-002D	44-45	RTHL*RTSH	MONITOR RETURN POINTER
002D	45	V2	LOW-RES COLOR GRAPHICS V2
002D	45	RPH2M	MONITOR MEMORY LOCATION 'RPH2M'
002D	45	V2	MONITOR MEMORY LOCATION 'V2'
002E	46	MASK	LOW-RES COLOR GRAPHICS MASK
002E	46	CHMSUM	MONITOR MEMORY LOCATION 'CHMSUM'
002E-002F	46-47	FORMAT	MONITOR 5 MINIASSEMBLER MEMORY LOCATION 'FORMAT'
002F	47	LASTIN	MONITOR MEMORY LOCATION 'LASTIN'
002F	47	LENGTH	MONITOR 5 MINIASSEMBLER MEMORY LOCATION 'LENGTH'
002F	47	SIGN	MONITOR MEMORY LOCATION 'SIGN'
0030	48	COLOR	LOW-RES COLOR GRAPHICS COLOR (FOR PLOT/HLIN/VLIN FUNCTIONS)
0030	48	HMASK	HI-RES GRAPHICS HMASK ON-THE-FLY DIT MASK
0031	49	MODE	MONITOR 5 MINIASSEMBLER MEMORY LOCATION 'MODE'
0032	50	INFLG	VIDEO FORMAT CONTROL (2510FF+NORMAL.127(47F)+FLASHING: 6310FF+INVL)
0033	51	PROMPT	PROMPT CHARACTER PRINTED ON GETLN CALL
0034	52	YSAV	MONITOR 5 MINIASSEMBLER MEMORY LOCATION 'YSAV'
0035	53	YSAV1	MONITOR MEMORY LOCATION 'YSAV1'
0035	53	L	MINIASSEMBLER MEMORY LOCATION 'L'
0036-0037	54-55	CSML*CSMH	PROGRAM COUNTER FOR USER EXIT ON CURT ROUTINE (MONITOR)
003B-0039	56-59	KSM*KSMM	PROGRAM COUNTER FOR USER EXIT ON KEYIN ROUTINE (MONITOR)
003A-003B	58-59	PCL*PCH	USER PROGRAM COUNTER SAVED HERE ON DRK TO MONITOR
003C	60	X0T	MONITOR MEMORY LOCATION 'X0T'
003C	60	XG2N2	MONITOR MEMORY LOCATION 'XG2N2'
003C-003D	60-61	A1L-A1H	MONITOR WORK BYTE PAIR A1
003E-003F	62-63	A2L-A2H	MONITOR WORK BYTE PAIR A2
0040-0041	64-65	A3L-A3H	MONITOR WORK BYTE PAIR A3
0042-0043	66-67	A4L-A4H	MONITOR WORK BYTE PAIR A4
0044	68	FMT	MINIASSEMBLER MEMORY LOCATION 'FMT'
0044-0045	68-69	A5L-A5H	MONITOR WORK BYTE PAIR A5
0045	69	ACC	USER AC SAVED HERE ON DRK TO MONITOR
0046	70	IREG	USER X-REG SAVED HERE ON DRK TO MONITOR
0047	71	YREG	USER Y-REG SAVED HERE ON DRK TO MONITOR
0048	72	STATUS	USER P STATUS SAVED HERE ON DRK TO MONITOR
0049	73	SPNT	USER STACK POINTER SAVED HERE ON DRK
004A-004B	74-75	L0HEM*L0HEMH	POINTER TO L0HEM
004C-004D	76-77	H1HEM*H1HEMH	POINTER TO H1HEM
004E-004F	78-79	RNDL*RNDH	16 DIT NO. RANDOMIZED WITH EACH KEY ENTRY
0050-0061	80-97		GENERAL PURPOSE POINTERS FOR APPLESOFT
0050-0051	80-81	ACL*ACH	MONITOR POINTER 'AC'
0050-0051	80-81	DIL*DIH	HI-RES GRAPHICS DELTA-Y FOR HLIN SHAPE
0051	82	SHAPEX	HI-RES GRAPHICS SHAPE XEMP
0052	82	DY	HI-RES GRAPHICS DELTA-Y FOR HLIN SHAPE
0052-0053	82-83	XINDL*XINDH	MONITOR 16-DIT POINTER 'XIND'
0053	83	QDRNT	HI-RES GRAPHICS QDRNT 2 L5D 5 ARE ROTATION QUADRANT FOR DRAM
0054	84	EL	HI-RES GRAPHICS ERROR FOR HLIN
0054-0055	84-85	AUXL*AUXH	MONITOR 16-DIT POINTER 'AUX'
0055-0056	84-85	EL*EH	HI-RES GRAPHICS ERROR FOR HLIN
0055	85	EH	HI-RES GRAPHICS ERROR FOR HLIN
0062-0066	90-102		RESULT OF LAST MULTIPLY/DIVIDE
0067-0068	103-104	START PRDC PTR	POINTER TO BEGINNING OF PROGRAM NORMALLY \$0801
0069-006A	105-106	L0HEM	POINTER TO START OF SIMPLE VARIABLE SPACE
006B-006C	107-108	ARAYV POINTER	POINTER TO BEGINNING OF ARAY SPACE
006D-006E	109-110	FREE SPACE PNTR	POINTER TO END OF NUMERIC STORAGE IN USE
006F-0070	111-112	STRING POINTER	POINTER TO START OF STRING STORAGE STRINGS TO END OF MEMORY

LUEBBERT'S COPYRIGHT APPLE MEMORY ATLAS - MICRO MAGAZINE VERSION

HEXLOC	DECLOC	NAME	USE
0083-0084	131-132		POINTER TO THE LAST-USED VARIABLE'S VALUE
0085-009C	133-136		GENERAL USAGE
009D	137	PICK	MONITOR MEMORY LOCATION 'PICK'
009D-00A3	137-163		MAIN FLOATING-POINT ACCUMULATOR *
00A4	164		GENERAL USE IN FLOATING POINT MATH ROUTINES
00A5-00AB	165-171		SECONDARY FLOATING POINT ACCUMULATOR
00AC-00AE	172-174		GENERAL USAGE FLAGS/POINTERS
00AF-00B0	175-176	PROGRAM POINTER	POINTER TO END OF PROGRAM NOT CHANGED BY L0HEM
00B1	177		CHRGCT 8/R CALL - GETS NEXT SEQUENTIAL CHR OR TOKEN
00B1-00CB	177-200		CHRGCT ROUTINE CALLED WHEN A-S WANTS ANOTHER CHARACTER
00B7	183	CHROOT	CHRGCT 8/R CALL CHRGCT INCREMENTS IXTRP CHRGCT DOES NOT
00B8-00B9	184-185		PTR TO LAST CHAR OBTAINED THRU CHRGCT ROUTINE
00B9-00C9	186-203	IXTRP	IXTRP - POINTS AT NEXT CHAR OR TOKEN FROM PROG (C/A DEC 78)
00C9-00CD	201-205		RANDOM NUMBER
00CA-00CC	202-203	PPL*PPH	BASIC START-OF-PROGRAM POINTER
00CC-00CD	204-205	PVL*PVH	BASIC END OF VARIABLES POINTER
00CE-00CF	206-207	ACL*ACH	BASIC ACC
00D0-00DF	216-223		DMERR POINTERS/SCRATCH
00DB-00D9	184-185		POINTER TO TOCLEAR ERROR FLAG
00E9-00ED	221-225		WHEN ERROR OCCURS, Y-ERR CODE APPEARS HERE
00EE	222		HI-RES GRAPHICS ILY COORDINATES
00ED-00EE	224-226		HI-RES GRAPHICS COLOR BYTE
00EE	228		GENERAL USAGE FOR HI-RES GRAPHICS
00E5-00E7	229-231		POINTER TO BEGINNING OF SHAPE TABLE
00E8-00E9	232-233		COLLISION COUNTER FOR HI-RES GRAPHICS
00EA	234		GENERAL USE FLAGS
00F0-00F3	240-243		MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'SIGN'
00F3	244	SIGN	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'SIGN'
00F4	244	X2	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'X2' (EXPONENT 2)
00F4-00FB	244-248		DMERR POINTERS
00F7	245	H2	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'H2' (MANTISSA 2)
00F7	247	SIAPAG	SHEET-16 MEMORY LOCATION 'SIAPAG'
00F8	248	X1	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'X1' (EXPONENT 1)
00F9	249	M1	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'M1' (MANTISSA 1)
00FC	252	E	MONITOR & FLOATING POINT ROUTINES MEMORY LOC 'E'
0010-001FF	256-511		SUBROUTINE RETURN STACK
0020-002FF	512-767	IN	MONITOR 5 MINIASSEMBLER MEMORY LOCATION 'IN'
0030-003FF	768-1023		KEYIN (INPUT) BUFFER
0030-003F	768-1015		AREA CLODDERED BY EITHER MASTER OR SLAVE DISKETTE BOOT
0030-003AF	768-943		OFTEN FREE SPACE NOTE COMPETING USES OFTEN FREE SPACE CONSTRAINTS
0030-00321	1004-801		DECRITER PRINTER OUTPUT (IF BLADED FROM DISK)
0032	802	XOL*XOH	HI-RES GRAPHICS XOL - XCOORD SAVED AFTER HLIN OR PLOT
0032	803	Y0	HI-RES GRAPHICS Y0 - MOST RECENT Y-COORDINATE
0032	804	BISAV	HI-RES GRAPHICS 'BISAV'
0032A	803	HCOLOR	HI-RES GRAPHICS COLOR FOR PLOT* HPOSN
0032B	805	MNDX	HI-RES GRAPHICS MNDX - ON-THE-FLY BYTE INDEX FROM BASE ADDRESS
0032E	806	HPAG	POKE 32 FOR HI-RES PLOT PLOTTING 64 FOR PAGE2
0032F	806	HPAG	HI-RES GRAPHICS HPAG - FOR PLOTTING GRAPHICS \$20 FOR PG1 \$40 FOR PG2
00327	807	SCALE	ON-THE-FLY SCALE FACTOR FOR DRAM SHAPE* MOVE
0032B-00329	808-809	SHAPXL*SHAPXH	START-OF-SHAPE-TABLE POINTER
0032A	810	COLLSN	COLLISION COUNT FROM DRAM/DRAM1
003D0	976		DOB RE-ENTRY POINT (3D00)
003D0	976		INITIALIZE OR RE-INITIALIZE DOS (3D00)
003D3	979		DOS 3 1 HARD ENTRY POINT
003D6	982		DOS 3 1 ENTRY POINT FOR I/O PACKAGE
003D9	985		DOS 3 1 ENTRY POINT FOR RWTS
003DC	988		DOS 3 1 ENTRY POINT TO LOAD Y*A WITH ADDRESS AT END OF SYS DUFFER
003E3	993		DOS 3 1 ENTRY POINT TO LOAD Y*A WITH ADDRESS OF IOBLK
003EA	1002	1002	DOS 3 2 ENTRY POINT FOR ROUTINE THAT UPDATES I/O HOOK TABLES
003FB	1016	USRADR	CTL-Y WILL CAUSE JSR HERE
003FE	1019	NM1	NM1'S VECTORED TO THIS LOCATION
003FE	1022	IRADR	MONITOR MEMORY LOCATION 'IRADR'
003FE-003FF	1022-1023		IRG'S VECTORED TO ADDRESS WHOSE POINTER IS HERE
00400-007FF	1024-2043		SCREEN BUFFER (HARDWARE PAGES 4-7) (LOW-RES GRAPHICS & TEXT PAGE 1)
00478*5	1144*5	BRATE	SERIAL INTERFACE BAUD QUANTUM RATE 51* 19200 BAUD, 64*300 BAUD
00478*5	1144*5		SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #5
00478*5	1272*5	STBITS	SERIAL INTERFACE CONTAIN NUMBER OF STOP BITS (INCLUDING 1 PARITY BIT)
00478*5	1272*5		SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #5
00578*5	1400*5	STATUS	SERIAL INTERFACE PARITY CHECKSUM OPTIONS (SEE MANUAL)
00578*5	1400*5		SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOTS
00578*5	1528*5		SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #5
00678*5	1656*5	DYTE	SERIAL INTERFACE INPUT OUTPUT BUFFER
00678*5	1656*5		SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #5
00678*5	1784*5		SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #5
00678*5	1784*5	PHDTH	SERIAL INTERFACE PRINT LINE WIDTH (8 CHARS PER LINE)
00778*5	1912*5	NDITS	SERIAL INTERFACE NUMBER OF DATA BITS PLUS 1 FOR START BIT
00778*5	1912*5		SCRATCHPAD MEMORY BYTE FOR PERIPHERAL IN SLOT #5
00778*5	2040*5	FLAGS	SERIAL INTERFACE OPERATION MODE
00778*5	2040*5		INTERRUPT RETURN MEMORY BYTE FOR PERIPHERAL IN SLOT #5
00800	2048		DEFAULT INTEGER BASIC L0HEM
00800-009FF	2048-2559		AREA CLODDERED BY EITHER MASTER OR SLAVE DISKETTE BOOT
00800-008FF	2048-3071		SECONDARY SCREEN BUFFER (TEXT & LOW-RES GRAPHICS PAGE 2)

## LUEBBERT'S COPYRIGHT APPLE MEMORY ATLAS - MICRO MAGAZINE VERSION

HEXLOC	DECLOC	NAME	USE
00800-0C000	2048-49132		RANGE OF POSSIBLE SETTINGS FOR HIMEM (DEPENDING UPON MEM SIZE) DOS
00800-LOMEM	2048-LOMEM		PROGRAM STORAGE FOR ROM VERSION OF APPLESOFT
00C00	3072		DEFAULT LOCATION FOR START OF SHAPE TABLE AS SET BY HI-RES SHAPE LOAD
00C00-01FFF	3072-8191		OFTEN FREE SPACE
00CF2	3314		TO CNVRT A/S PROG FM ROM TO CASSETTE LOAD PROG CALL 3314*LIST*SAVE
01800-03FFF	4000-16383		THIS REGION OF MEMORY IS CLOBBED BY A SLAVE DISKETTE BOOT
01800-04000	6912-16384		RANDOS (VERSION OF DOS USED WITH MASTER CREATE - FROM DISK)
02000-03FFF	8192-16383		HI-RES GRAPHICS PAGE 1
03000-LOMEM	12288-LOMEM		PROGRAM STORAGE FOR RAM VERSION OF APPLESOFT
03F3-03F4	1011-1012		DOS 3 I - POKER TO ZEROS TO REDDIT HELLO PROGRAM
04000-04520	16384-17696		NORMAL LOCATION FOR MAPOR'S HI RES TEXT SET
04000-05FFF	16384-24575		HI-RES GRAPHICS PAGE 2
04500	17680		CALL FOR INVERSION BY MAPOR'S ROUTINE
04500-4520	17684-17696		S/R W/ MAPOR'S HI-RES TEXT SET TO INVERT WHITE TO BLACK & VICEVERSA
05600-06000	22016-32768		DISK OPERATING SYSTEM (DISK 1)
05600-05853	27136-26541		DOS 3 I USER BUFFER #1
05600-05700	27136-26890		DOS 3 I USER BUFFER #1 DATA BUFFER
05701-05802	26879-26890		DOS 3 I USER BUFFER #1 - LIST OF SECTOR & TRACK NUMBERS USED
05801-05853	26433-26541		DOS 3 I USER BUFFER #1 - FILE NAME & MISC DATA
05D10-?	25328-?		STARTING ADDRESSES FOR VARIOUS DOS3 I TASKS
05D73-0A7DF	23229-23561		SYSTEM SECTION OF DOS 3 I
05D89	29139		INITIALIZE OR RE-INITIALIZE DOS
05E4D	23011		ROUTINE WHICH HANDLES DOS INPUT HOOK
05E7E	24982		ROUTINE WHICH HANDLES DOS OUTPUT HOOK
0A184	24140		ADDRESS FOR DOS 3 I PRG COMMAND
0A189	24139		ADDRESS FOR DOS 3 I INR COMMAND
0A18E	24130		ADDRESS FOR DOS 3 I MON COMMAND
0A18C	24100		ADDRESS FOR DOS 3 I RAFILES COMMAND
0A18E	24082		ADDRESS FOR DOS 3 I DELETE COMMAND
0A18C	24068		ADDRESS FOR DOS 3 I LOCK COMMAND
0A200	24064		ADDRESS FOR DOS 3 I DSAVE COMMAND
0A200	24064		ADDRESS FOR DOS 3 I UNLOCK COMMAND
0A20B	24056		ADDRESS FOR DOS 3 I VERIFY COMMAND
0A20C	24052		ADDRESS FOR DOS 3 I REFAVE COMMAND
0A223	24049		ADDRESS FOR DOS 3 I APPEND COMMAND
0A23A	24010		ADDRESS FOR DOS 3 I OPEN COMMAND
0A270	23944		ADDRESS FOR DOS 3 I CLOSE COMMAND
0A2EC	23828		ADDRESS FOR DOS 3 I DLOAD COMMAND
0A327	23769		ADDRESS FOR DOS 3 I DRUM COMMAND
0A330	23760		ADDRESS FOR DOS 3 I SAVE COMMAND
0A345	23643		ADDRESS FOR DOS 3 I LOAD COMMAND
0A476	23434		ADDRESS FOR DOS 3 I RUN COMMAND
0A48D	23411		ADDRESS FOR DOS 3 I CHAIN COMMAND
0A4A5	23387		ADDRESS FOR DOS 3 I WRITE COMMAND
0A480	23376		ADDRESS FOR DOS 3 I READ COMMAND
0A4E4	23324		ADDRESS FOR DOS 3 I INIT COMMAND
0A351	23295		ADDRESS FOR DOS 3 I HNGON COMMAND
0A30D	23283		ADDRESS FOR DOS 3 I FP COMMAND
0A331	23247		ADDRESS FOR DOS 3 I INT COMMAND
0A34F	23217		ADDRESS FOR DOS 3 I EXEC COMMAND
0A366	23210		ADDRESS FOR DOS 3 I POSITION COMMAND
0A7E0-0A863	22960-22439		DOS COMMAND TABLE
0A8CD-0A980	22323-22144		DOS ERROR MSG TABLE
0A996-0A997	22122-22121		DOS INTERNAL HOOK ADDRESS TO OUTPUT A CHARACTER
0A998-0A999	22120-22119		DOS INTERNAL HOOK ADDRESS TO INPUT A CHARACTER
0A9A3-0A9A4	22109-22108		LENGTH OF DLOADED FILE
0A9B3-0A9B6	22091-22090		STARTING ADDRESS OF DLOADED FILE
0AAB8	22005		START OF LIST OF POINTERS TO SECTIONS OF DOS 3 I I/O PACKAGES
0AABF-0B2CE	21953-19762		DOS 3 I I/O PACKAGE
0B2EF-0B642	19473-18878		DOS 3 I SYSTEM BUFFER (FOR CATALOG ETC)
0BDD0	17152		ROUTINE WHICH READS IN DIRECTORY OFF DISK
0BF6	16304		VOL NO OF CURRENT DISK
0BFF	16384		HIGHEST RAM MEMORY ADDRESS
0C000	16384		DEFAULT INTEGER BASIC HIMEM (M/O DOS 3 48K MACHINE)
0C000-0C00F	16384-16369	KDD ~ IOADR	READ KEYBOARD IF VAL>127 THEN KEY WAS PRESSED
0C000-0CFFF	16384-12289		KEYBOARD INPUT SUBROUTINE
0C010	16368	NBDBTB	ADDRESSES DEDICATED TO HARDWARE FUNCTION
0C010-0C01F	16368-16353		CLEAR KEYBOARD STROBE POKER 0 ALWAYS AFTER READING KDD
0C020	16352		CLEAR KEYBOARD STROBE SUBROUTINE
0C021	16352		MONITOR MEMORY LOCATION 'TAPEOUT'
0C020	16352		TOGGLE CASSETTE OUTPUT
0C030	16336	SPKR	PEEK TO TOGGLE SPEAKER
0C041	16320		OUTPUT STROBE TO GAME I/O CONNECTOR
0C050	16304	TXICLR	POKER 0 TO SET GRAPHICS MODE
0C051	16303	TXIBET	POKER 0 TO SET TEXT MODE
0C052	16302	MXICLR	POKER 0 TO SET BOTTOM 4 LINES TO GRAPHICS
0C053	16301	MXIBET	POKER 0 TO SELECT TEXT/GRAPHICS MIX (BOTTOM 4 LINES TEXT)
0C054	16300	LOWSCR	POKER 0 TO DISPLAY PRIMARY PAGE (PAGE 1)
0C055	16299	HISCR	POKER 0 TO DISPLAY SECONDARY PAGE (PAGE 2)
0C056	16298	LOWES	POKER 0 TO SET LO-RES GRAPHICS
0C057	16297	HIREG	POKER 0 TO SET HI-RES GRAPHICS

## LUEBBERT'S COPYRIGHT APPLE MEMORY ATLAS - MICRO MAGAZINE VERSION

HEXLOC	DECLOC	NAME	USE
0C058	16296		POKER 0 TO CLEAR GAME I/O OUTPUT AND
0C059	16295		POKER 0 TO SET GAME I/O OUTPUT AND
0C05A	16294		POKER 0 TO CLEAR GAME I/O OUTPUT AND
0C05B	16293		POKER 0 TO SET GAME I/O OUTPUT AND
0C05C	16292		POKER 0 TO CLEAR GAME I/O OUTPUT AND
0C05D	16291		POKER 0 TO SET GAME I/O OUTPUT AND
0C05E	16290		POKER 0 TO CLEAR GAME I/O OUTPUT AND
0C05F	16289		POKER 0 TO SET GAME I/O OUTPUT AND
0C060	16288	TAPEIN	MONITOR MEMORY LOCATION 'TAPEIN'
0C060/B	16288		STATE OF 'CASSETTE DATA IN' APPEARS IN DIT 7
0C061	16287		PEEK TO READ PDL(1) IF '127 SWITCH ON
0C062	16286		PEEK TO READ PDL(1) PUSH BUTTON SWITCH
0C06B	16285		PEEK TO READ PDL(2) PUSH BUTTON SWITCH
0C064	16188	PADDLO	MONITOR MEMORY LOCATION PADDLO
0C064/C	16188		STATE OF TIMER OUTPUT FOR PADDLE 1 APPEARS IN DIT 7
0C065/D	16187		STATE OF TIMER OUTPUT FOR PADDLE 2 APPEARS IN DIT 7
0C06A/E	16186		STATE OF TIMER OUTPUT FOR PADDLE 3 APPEARS IN DIT 7
0C067/F	16185		STATE OF TIMER OUTPUT FOR PADDLE 4 APPEARS IN DIT 7
0C070	16272	PTRG	MONITOR MEMORY LOCATION 'PTRIG' (PADDLE TRIGGER)
0C071	16272	PRIG	TRIGGER PADDLE TIMERS DURING PHIG-
0C081	16256		DEVICE SELECT 0
0C08X	16240		DEVICE SELECT 1
0C08Y	16131	DEVICE SELECT 2	DEVICE SELECT 2
0C08Z	16208		DEVICE SELECT 3
0C081	16192		DEVICE SELECT 4
0C0D1	16176		DEVICE SELECT 5
0C0EB	16152		ADDRESS TO POWER DOWN DISK IN SLOT 6
0C0F9	16131		ADDRESS TO POWER UP DISK IN SLOT 6
0C0E1	16160		DEVICE SELECT 6
0C0F1	16144		DEVICE SELECT 7
0C100	16128		CALL -16128 IS EQUIVALENT TO P#81 FOR INITIALIZING SERIAL INTERFACE
0C100	16128		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #1
0C200	15842		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #2
0C300	15616		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #3
0C400	15360		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #4
0C500	15104		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #5
0C600	14848		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #6
0C700	14592		STANDARD CHARACTER I/O SUBROUTINE ENTRY POINT FOR SLOT #6
0C800-0CFFF	14336-12289		PIN 20 ON ALL PERIPH CONTS GOES LOW DURING PHIO ON READ OR WRITE
0C900	14105		SERIAL INTERFACE BATCH INPUT ROUTINE - A1&A2 SPECIFY MEMORY RANGE
0C911	14105		SERIAL INTERFACE BATCH OUTPUT ROUTINE - A1&A2 SPECIFY MEMORY RANGE
0C900	16204+236+8		TRANSMIT ASCII CHAR IN ACCUMULATOR OUT VIA SERIAL INTERFACE IN SLOT 5
0D000	12288	SETHRL	HI-RES GRAPHICS INIT S/R CALL (ROM VERSION)
0D000-0D3FF	12288-11265		HI-RES GRAPHICS ROM
0D000-0D7FF	12200-10241		ROM SOCKET D0
0D00E	12274	HCLR	HI-RES GRAPHICS CLEAR S/R CALL
0D010	12272	BKNGND	HI-RES GRAPHICS 'BKNGND (HOLD) SET FOR BLACK BKNGND
0D012	12270	BKNGD	HI-RES GRAPHICS MEMORY LOCATION 'BKNGD' (ROM)
0D01C	11780		HI-RES GRAPHICS FIND S/R CALL PARAM=SHAPE*ROT*SCALE
0D0F7	11527		HI-RES GRAPHICS PDBN S/R CALL PARAM=X*Y*COLR
0D00E	11506		HI-RES GRAPHICS PLOT S/R CALL PARAM=X*Y*COLR
0D314	11500		HI-RES GRAPHICS LINE S/R CALL PARAM=X*Y*COLR
0D331	11471		HI-RES GRAPHICS BKNGD S/R CALL PARAM=COLR
0D337	11465		HI-RES GRAPHICS LINE S/R CALL PARAM=X*Y*COLR
0D34C	11462		HI-RES GRAPHICS DRAWI S/R CALL PARAM=X*Y*COLR*SHAPE*ROT*SCALE
0D389	11335		HI-RES GRAPHICS SHLQAD S/R CALL
0D40C	11076		INTEGER BASIC PARI APPEND PROGRAM ENTRY
0D4F2	11022		TO CONVERT A/S FM CASSETTE TO ROM- LD FM CAS*CALL -11022*LIST*SAVE
0D535	10955		INTEGER BASIC PARI TAPE VERIFY PROG ENTRY
0D6DD	10531		INTEGER BASIC PARI REMUMBER PROG ENTRY (WHOLE PROG)
0D6E7	10521		INTEGER BASIC PARI REMUMBER PROG ENTRY (PARI PROG)
0D717	10473		INTEGER BASIC PARI MUSIC PROG ENTRY
0D800-0DFFF	10240-8193		ROM SOCKET D8
0D867	8867		FRMNUM S/R EVALS FORMULA EXP INTO FLOATING PT ACCUR
0D8C9	8503		SNERR S/R PRINTS 'SYNTAX ERROR' AND HALTS PROG
0E000	8192	DASIC	ENTER INTEGER BASIC
0E000-0E7FF	8192-6145		ROM SOCKET E0 (INTEGER BASIC)
0E003	8189	DASIC2	ENTRY 2 OF INTEGER BASIC
0E268	7317	MEMFUL	INTEGER BASIC MEMORY FULL ERROR
0E318	6885		INTEGER BASIC DECIMAL LPRINT S/R
0E40B	6408		GETBYT S/R EVALS FORMULA 1 (CONVTS TO 1-BYT VAL IN X REG
0E800-0EFFF	6144-4097		ROM SOCKET E8 (INTEGER BASIC)
0E840	4504	RNGERR	INTEGER BASIC RANGE ERROR
0F000-0F7FF	4096-2049		ROM SOCKET F0 (1K INTEGER BASIC ~ 1 K MONITOR)
0F11E	3810	ACADR	HI-RES GRAPHICS 2-BYTE TAPE READ SETUP
0F666	2458		TURN ON MINIASEMBLER
0F6D9	2423		SHEET-16 INTERPRETER ENTRY
0F800	2048	PLOT	MONITOR S/R PLOT A POINT (LO-RES) AC Y-COORD Y X-COORD
0F809	2048	PLOT	MONITOR S/R PLOT A POINT (HI-RES) AC Y-COORD Y X-COORD
0F800-0FFFF	2048-1		ROM SOCKET F8 (MONITOR)

HEXLOC	DECLOC	NAME	USE
9F80C	-2036	RTHASK	MONITOR MEMORY LOCATION 'RTHASK'
9F80E	-2034	PLOT1	MONITOR MEMORY LOCATION 'PLOT1'
9F819	-2033	HLINE	HLINE S/R (SEE CALL-APPLE NOV/DEC 78 PG4)
9F819	-2023	HLINE	MONITOR S/R TO DRAW A HORIZONTAL LINE (LO-RES)
9F81C	-2020	HLINE1	MONITOR MEMORY LOCATION 'HLINE1'
9F826	-2010	VLINEX	MONITOR MEMORY LOCATION 'VLINEX'
9F828	-2008	VLINE	DRAW A VERTICAL LINE
9F831	-1999	RTS1	MONITOR MEMORY LOCATION 'RTS1'
9F832	-1998	CLRSCL	CLEAR SCREEN GRAPHICS MODE
9F832	-1998	CLRSCL	CLEAR LOW RES GRAPHICS SCREEN1
9F838	-1994	CLRTOP	MONITOR MEMORY LOCATION 'CLRTOP'
9F838	-1992	CLRSCL2	MONITOR MEMORY LOCATION 'CLRSCL2'
9F83C	-1988	CLRSCL3	MONITOR MEMORY LOCATION 'CLRSCL3'
9F847	-1977	QBASCALC	MONITOR S/R TO CALCULATE GRAPHICS BASE ADDRESS
9F856	-1962	QBASCALC	MONITOR MEMORY LOCATION 'QBASCALC'
9F85F	-1953	NITCOL	MONITOR S/R - INCREMENT COLOR BY 3
9F864	-1948	SETCOL	MONITOR S/R TO ADJUST COLOR BYTE FOR BOTH HALVES EQUAL
9F871	-1935	SCRN	SCRN S/R (LO-RES GRAPHICS) (SEE CALL-APPLE DEC78)
9F871	-1935	SCRN	MONITOR S/R TO GET SCREEN COLOR
9F879	-1927	SCRN2	MONITOR MEMORY LOCATION 'SCRN2'
9F87F	-1921	RTHSK2	MONITOR MEMORY LOCATION 'RTHSK2'
9F882	-1918	INSDS1	MONITOR MEMORY LOCATION 'INSDS1'
9F88E	-1906	INSDS2	MONITOR S/R - DISASSEMBLER ENTRY
9F898	-1893	IEVEN	MONITOR MEMORY LOCATION 'IEVEN'
9F8A5	-1893	ERR	MONITOR MEMORY LOCATION 'ERR'
9F8A9	-1879	GETFHT	MONITOR MEMORY LOCATION 'GETFHT'
9F8DE	-1858	MNNDX1	MONITOR MEMORY LOCATION 'MNNDX1'
9F8E2	-1834	MNNDX2	MONITOR MEMORY LOCATION 'MNNDX2'
9F8C9	-1847	MNNDX3	MONITOR MEMORY LOCATION 'MNNDX3'
9F8D0	-1840	INSTDSP	MONITOR & MINIASSEMBLER MEMORY LOCATION 'INSTDSP'
9F8D4	-1836	PRINTOP	MONITOR MEMORY LOCATION 'PRINTOP'
9F8D8	-1829	PRINTDL	MONITOR MEMORY LOCATION 'PRINTDL'
9F8F3	-1803	PRPNI	MONITOR MEMORY LOCATION 'PRPNI'
9F8F9	-1799	PRPN2	MONITOR MEMORY LOCATION 'PRPN2'
9F910	-1776	PRADR1	MONITOR MEMORY LOCATION 'PRADR1'
9F912	-1772	PRADR2	MONITOR MEMORY LOCATION 'PRADR2'
9F926	-1754	PRADR3	MONITOR MEMORY LOCATION 'PRADR3'
9F92A	-1750	PRADR4	MONITOR MEMORY LOCATION 'PRADR4'
9F930	-1744	PRADR5	MONITOR MEMORY LOCATION 'PRADR5'
9F938	-1736	RELADR	MONITOR MEMORY LOCATION 'RELADR'
9F940	-1728	PRINTYX	MONITOR S/R - PRINT CONTENTS OF Y AND X AS 4 HEX DIGITS
9F941	-1727	PRINTAX	MONITOR MEMORY LOCATION 'PRINTAX'
9F944	-1724	PRINTX	MONITOR MEMORY LOCATION 'PRINTX'
9F948	-1720	PRDLNK	MONITOR MEMORY LOCATION 'PRDLNK'
9F94C	-1716	PRDL2	MONITOR S/R - PRINT DLANKS X REG CONTAINS NUMBER TO PRINT
9F94C	-1716	PRDL3	MONITOR MEMORY LOCATION 'PRDL3'
9F953	-1709	PCADJ	MINIASSEMBLER MEMORY LOCATION 'PCADJ'
9F953	-1708	PCADJ2	MONITOR & MINIASSEMBLER MEMORY LOCATION 'PCADJ2'
9F956	-1706	PCADJ4	MONITOR MEMORY LOCATION 'PCADJ4'
9F961	-1693	RTS2	MONITOR MEMORY LOCATION 'RTS2'
9F962	-1694	FNT1	MONITOR MEMORY LOCATION 'FNT1'
9F966	-1626	FNT2	MONITOR MEMORY LOCATION 'FNT2'
9F984	-1612	CHAR1	MONITOR & MINIASSEMBLER MEMORY LOCATION 'CHAR1'
9F984	-1612	CHAR2	MONITOR & MINIASSEMBLER MEMORY LOCATION 'CHAR2'
9F9C0	-1600	MNEFL	MONITOR & MINIASSEMBLER MEMORY LOCATION 'MNEFL'
9FA00	-1526	MNEHR	MONITOR & MINIASSEMBLER MEMORY LOCATION 'MNEHR'
9FA43	-1469	STEP	MONITOR S/R - PERFORM A SINGLE STEP
9FA4E	-1458	IOINIT	MONITOR MEMORY LOCATION 'IOINIT'
9FA78	-1416	X01	MONITOR MEMORY LOCATION 'X01'
9FA7A	-1414	X02	MONITOR MEMORY LOCATION 'X02'
9FA86	-1402	IRQ	MONITOR S/R - IRQ HANDLER
9FA92	-1390	DREAK	MONITOR S/R - BREAK HANDLER
9FA9C	-1380	XDRK	MONITOR MEMORY LOCATION 'XDRK'
9FAA3	-1371	XRT1	MONITOR MEMORY LOCATION 'XRT1'
9FAA3	-1367	XRT5	MONITOR MEMORY LOCATION 'XRT5'
9FAAD	-1363	PCINC2	MONITOR MEMORY LOCATION 'PCINC2'
9FAAF	-1361	PCINC3	MONITOR MEMORY LOCATION 'PCINC3'
9FAB9	-1331	XJSR	MONITOR MEMORY LOCATION 'XJSR'
9FAC4	-1340	XJMP	MONITOR MEMORY LOCATION 'XJMP'
9FAC4	-1339	XJMPAT	MONITOR MEMORY LOCATION 'XJMPAT'
9FACD	-1331	NEWPCFL	MONITOR MEMORY LOCATION 'NEWPCFL'
9FAD1	-1327	RTNJMP	MONITOR MEMORY LOCATION 'RTNJMP'
9FAD7	-1321	REGDSP	MONITOR S/R TO DISPLAY USER REGISTERS
9FADA	-1318	RGDSP1	MONITOR MEMORY LOCATION 'RGDSP1'
9FAE4	-1308	RGDSP2	MONITOR MEMORY LOCATION 'RGDSP2'
9FAED	-1303	BRANCH	MONITOR MEMORY LOCATION 'BRANCH'
9FBD0	-1269	NBRNCH	MONITOR MEMORY LOCATION 'NBRNCH'
9FBD1	-1263	INITDL	MONITOR MEMORY LOCATION 'INITDL'
9FD19	-1255	RTDL	MONITOR MEMORY LOCATION 'RTDL'
9FD1E	-1250	PREAD	MONITOR S/R TO READ PADDLE X-REG CONTAINS PADDLE NUMBER 0-3
9FD23	-1243	PREAD2	MONITOR MEMORY LOCATION 'PREAD2'

HEXLOC	DECLOC	NAME	USE
9FB2E	-1234	RTS2D	MONITOR MEMORY LOCATION 'RTS2D'
9FB2F	-1233	INIT	MONITOR S/R - SCREEN INITIALIZATION
9FB39	-1223	SETTAT	MONITOR S/R - SET SCREEN TO TEXT MODE
9FB40	-1216	SETOR	MONITOR S/R - SET GRAPHIC MODE (GR) CLODDERS ACCUMULATOR
9FB48	-1203	SETMND	MONITOR S/R - SET NORMAL WINDOW
9FD30	-1189	TADV	MONITOR MEMORY LOCATION 'TADV'
9FB60	-1184	MULPM	MONITOR MEMORY LOCATION 'MULPM'
9FB63	-1181	MUL	MONITOR S/R - MULTIPLY ROUTINE
9FB65	-1179	MUL2	MONITOR MEMORY LOCATION 'MUL2'
9FB6D	-1171	MUL3	MONITOR MEMORY LOCATION 'MUL3'
9FB76	-1162	MUL4	MONITOR MEMORY LOCATION 'MUL4'
9FB78	-1160	MUL5	MONITOR MEMORY LOCATION 'MUL5'
9FB81	-1131	DIVPM	MONITOR MEMORY LOCATION 'DIVPM'
9FB84	-1148	DIV	MONITOR S/R - DIVIDE ROUTINE
9FB86	-1146	DIV2	MONITOR MEMORY LOCATION 'DIV2'
9FB8A	-1120	DIV3	MONITOR MEMORY LOCATION 'DIV3'
9FB84	-1116	RD1	MONITOR MEMORY LOCATION 'RD1'
9FB8F	-1103	RD2	MONITOR MEMORY LOCATION 'RD2'
9FB84	-1100	RD3	MONITOR MEMORY LOCATION 'RD3'
9FB8C	-1089	RDRTS	MONITOR MEMORY LOCATION 'RDRTS'
9FB81	-1087	DASCALC	MONITOR S/R - CALCULATE TEXT BASE ADDRESS
9FB80	-1072	DSCLC2	MONITOR MEMORY LOCATION 'DSCLC2'
9FB82	-1063	BELL1	MONITOR MEMORY LOCATION 'BELL1'
9FB82	-1062	BELL2	MONITOR S/R - SOUND BELL (BEEPER)
9FB8F	-1041	RTS2D	MONITOR MEMORY LOCATION 'RTS2D'
9FBFO	-1040	STOADV	MONITOR MEMORY LOCATION 'STOADV'
9FBF4	-1026	ADVANCE	MONITOR S/R - MOVE CURSOR RIGHT
9FBFC	-1028	RTS3	MONITOR MEMORY LOCATION 'RTS3'
9FB87	-1037	WDOUT	MONITOR S/R - OUTPUT A REGISTER AS ASCII ON TEXT SCREEN 1
9FC10	-1008	DS	MONITOR S/R TO MOVE CURSOR LEFT (BACKSPACE)
9FC1A	-998	UP ~ CURSUP	MONITOR S/R TO CURSOR UP
9FC22	-990	VTAB	MONITOR S/R - PERFORM A VERTICAL TAB TO ROW SPECIFIED IN ACCUM (80-817)
9FC24	-988	VTAD2	MONITOR MEMORY LOCATION 'VTAD2'
9FC28	-981	RTS4	MONITOR MEMORY LOCATION 'RTS4'
9FC2C	-980	ESC1	MONITOR S/R - PERFORM ESCAPE FUNCTIONS
9FC42	-958	CLEDDP	MONITOR S/R TO CLEAR FROM CURSOR TO END OF PAGE CLODDERS ACC & Y-REG
9FC46	-934	CLEDP1	MONITOR MEMORY LOCATION 'CLEDP1'
9FC58	-936	HOME	MONITOR S/R TO HOME CURSOR & CLEAR SCREEN CLODDERS ACCUM & Y-REG
9FC62	-926	CF	MONITOR S/R TO PERFORM A CARRIAGE RETURN
9FC66	-922	LF	MONITOR S/R TO PERFORM A LINE FEED
9FC70	-912	SCROLL	MONITOR S/R TO SCROLL UP 1 LINE CLODDERS ACCUM & Y-REG
9FC76	-906	SCRL1	MONITOR MEMORY LOCATION 'SCRL1'
9FC8C	-884	SCRL2	MONITOR MEMORY LOCATION 'SCRL2'
9FC95	-873	SCRL3	MONITOR MEMORY LOCATION 'SCRL3'
9FC9C	-868	CLEDL	MONITOR S/R TO CLEAR TO END OF LINE
9FC9E	-866	CLEDL1	MONITOR MEMORY LOCATION 'CLEDL1'
9FCAD	-864	CLEDL2	MONITOR MEMORY LOCATION 'CLEDL2'
9FCAB	-856	WAIT	CALL FOR WAIT LOOP
9FC49	-833	WAIT2	MONITOR MEMORY LOCATION 'WAIT2'
9FC4A	-834	WAIT3	MONITOR MEMORY LOCATION 'WAIT3'
9FC84	-844	NXTA1	MONITOR S/R TO INCREMENT A1 (16 BITS) THEN DO NEXT1
9FC84	-838	NXTA1	MONITOR S/R TO INCREMENT A1 (16 BITS) SET CARRY IF RESULT >A2
9FC8B	-824	RTS4B	MONITOR MEMORY LOCATION 'RTS4B'
9FC99	-823	HEADR	MONITOR MEMORY LOCATION 'HEADR'
9FC86	-810	WRBIT	MONITOR MEMORY LOCATION 'WRBIT'
9FC8B	-805	ZERDLY	MONITOR MEMORY LOCATION 'ZERDLY'
9FC2E	-798	OHEDLY	MONITOR MEMORY LOCATION 'OHEDLY'
9FC33	-795	URTAPE	MONITOR MEMORY LOCATION 'URTAPE'
9FCEC	-798	RDDYTE	MONITOR MEMORY LOCATION 'RDDYTE'
9FCE2	-786	RDDYT2	MONITOR MEMORY LOCATION 'RDDYT2'
9FCFA	-774	RDDIT	MONITOR TWO-EDGE TAPE SENSE
9FCFD	-771	RDDIT	MONITOR MEMORY LOCATION 'RDDIT'
9FDDC	-736	RKEY	GET KEY INPUT FROM THE KEYBOARD CLODDERS ACC & Y-REG
9FD18	-741	KEYIN	MONITOR S/R - MONITOR KEYIN ROUTINE
9FD21	-735	KEYIN2	MONITOR MEMORY LOCATION 'KEYIN2'
9FD2F	-721	ESC	MONITOR MEMORY LOCATION 'ESC'
9FD33	-715	RDCCHAR	CALL TO READ KEY & PERFORM ESCAPE FUNCTION IF NECESSARY
9FD3D	-707	NOTICR	MONITOR MEMORY LOCATION 'NOTICR'
9FD3F	-673	NOTICR1	MONITOR MEMORY LOCATION 'NOTICR1'
9FD62	-670	CANCEL	MONITOR S/R TO PERFORM A LINE CANCEL (1)
9FD67	-665	GETLNZ	MONITOR S/R TO PERFORM CARRIAGE RETURN AND GET A LINE OF TEXT
9FD6A	-662	GETLN	MONITOR S/R TO GET LINE OF TEXT FROM KEYBD X RETND W/ # OF CHARS
9FD71	-655	BCKSPC	MONITOR MEMORY LOCATION 'BCKSPC'
9FD75	-631	NXTCHAR	MONITOR MEMORY LOCATION 'NXTCHAR'
9FD7E	-642	CANST	MONITOR MEMORY LOCATION 'CANST'
9FD80	-640	INSTDSP	MONITOR S/R TO DISASSEMBLE INSTRUCTION AT PCH/PCFL
9FD84	-636	ADDINP	MONITOR MEMORY LOCATION 'ADDINP'
9FD8E	-626	CRDUT	MONITOR S/R TO PRINT A CARRIAGE RETURN CLODDERS ACC & Y-REG
9FD92	-622	PRAI	MONITOR MEMORY LOCATION 'PRAI'
9FD96	-618	PRYX2	MONITOR MEMORY LOCATION 'PRYX2'

LUEBBERT'S COPYRIGHT APPLE MEMORY ATLAS - MICRO MAGAZINE VERSION

HEXLOC	DECLOC	NAME	USE
\$FDA3	-403	IAMB	MONITOR MEMORY LOCATION 'IAMB'
\$FDAD	-593	H0DBCHM	MONITOR MEMORY LOCATION 'H0DBCHM'
\$FDB3	-589	IAM	MONITOR MEMORY LOCATION 'IAM'
\$FDB6	-586	DATAOUT	MONITOR MEMORY LOCATION 'DATAOUT'
\$FDC3	-571	RTS4C	MONITOR MEMORY LOCATION 'RTS4C'
\$FDC5	-570	IAMPH	MONITOR MEMORY LOCATION 'IAMPH'
\$FDD1	-559	ADD	MONITOR MEMORY LOCATION 'ADD'
\$FDDA	-550	PRBYTE	MONITOR S/R TO PRINT CONTENTS OF ACC AS 2 HEX DIGITS
\$FDE3	-541	PRHEX	MONITOR S/R TO PRINT A HEX DIGIT
\$FDE5	-539	PRHEXZ	MONITOR MEMORY LOCATION 'PRHEXZ'
\$FDEB	-531	COUT	MONITOR S/R TO OUTPUT CHAR IN ACC CLOSERS ACC-Y-REG-COUT
\$FDF0	-528	COUT1	MONITOR S/R TO GET MONITOR CHARACTER OUTPUT
\$FDF6	-522	COUTZ	MONITOR MEMORY LOCATION 'COUTZ'
\$FE00	-512	BL1	MONITOR & MINIASSEMBLER MEMORY LOCATION 'BL1'
\$FE04	-508	BLANK	MONITOR MEMORY LOCATION 'BLANK'
\$FE0B	-501	STOR	MONITOR MEMORY LOCATION 'STOR'
\$FE17	-489	RTS5	MONITOR MEMORY LOCATION 'RTS5'
\$FE1B	-488	SETHODE	MONITOR MEMORY LOCATION 'SETHODE'
\$FE1D	-483	SETHDZ	MONITOR MEMORY LOCATION 'SETHDZ'
\$FE20	-480	LT	MONITOR MEMORY LOCATION 'LT'
\$FE22	-478	LTZ	MONITOR MEMORY LOCATION 'LTZ'
\$FE2C	-468	MOVE	MONITOR S/R TO PERFORM A MEMORY MOVE (A1-A2 TO A4)
\$FE36	-458	VFY	MONITOR S/R TO PERFORM A MEMORY VERIFY
\$FE5B	-424	VFYOK	MONITOR MEMORY LOCATION 'VFYOK'
\$FE5E	-418	LIST	CALL TO DISASSEMBLE 20 INSTRUCTIONS
\$FE63	-410	LIST2	MONITOR MEMORY LOCATION 'LIST2'
\$FE7B	-392	AIPCLP	MONITOR & MINIASSEMBLER MEMORY LOCATION 'AIPCLP'
\$FE7E	-385	AIPCTS	MONITOR MEMORY LOCATION 'AIPCTS'
\$FE80	-384	SETINV	MONITOR MEMORY LOCATION 'SETINV'
\$FE84	-380	SETNORM	MONITOR MEMORY LOCATION 'SETNORM'
\$FE86	-378	SETIFLO	MONITOR MEMORY LOCATION 'SETIFLO'
\$FE89	-375	SETMBD	MONITOR MEMORY LOCATION 'SETMBD'
\$FE8B	-370	INPRT	MONITOR MEMORY LOCATION 'INPRT'
\$FE8D	-371	INPRT	MONITOR MEMORY LOCATION 'INPRT'
\$FE93	-365	SETVID	MONITOR MEMORY LOCATION 'SETVID'
\$FE95	-363	OUTPORT	MONITOR MEMORY LOCATION 'OUTPORT'
\$FE97	-361	OUTPRT	MONITOR MEMORY LOCATION 'OUTPRT'
\$FE9B	-357	IDPRT	MONITOR MEMORY LOCATION 'IDPRT'
\$FEA7	-345	IDPRT1	MONITOR MEMORY LOCATION 'IDPRT1'
\$FEA9	-343	IDPRT2	MONITOR MEMORY LOCATION 'IDPRT2'
\$FEB0	-336	IBASIC	MONITOR S/R TO JUMP TO BASIC
\$FEB3	-333	BASCONT	MONITOR S/R TO CONTINUE BASIC
\$FEB6	-330	GO	MONITOR MEMORY LOCATION 'GO'
\$FEBF	-321	REGZ	MONITOR MEMORY LOCATION 'REGZ'
\$FEC2	-318	TRACE	CALL TO PERFORM MONITOR TRACE
\$FEC4	-316	STEPZ	MONITOR MEMORY LOCATION 'STEPZ'
\$FECA	-310	USR	MONITOR MEMORY LOCATION 'USR'
\$FECD	-307	WRITE	MONITOR S/R TO WRITE TO CASSETTE TAPE
\$FEDA	-300	WRI	MONITOR MEMORY LOCATION 'WRI'
\$FEE0	-275	WRBYTE	MONITOR MEMORY LOCATION 'WRBYTE'
\$FEE2	-273	WRBYT2	MONITOR MEMORY LOCATION 'WRBYT2'
\$FEE6	-266	CRMON	MONITOR MEMORY LOCATION 'CRMON'
\$FEDD	-259	READ	CALL TO READ FROM TAPE - LIMITS A1 & A2
\$FF02	-254	READX1	HI-RES GRAPHICS - READ WITHOUT HEADER
\$FF0A	-246	RDZ	MONITOR MEMORY LOCATION 'RDZ'
\$FF16	-234	RDZ	MONITOR MEMORY LOCATION 'RDZ'
\$FF2D	-211	PRERR	MONITOR S/R TO PRINT 'ERR' AND SOUND BELL CLOSERS ACC & Y-REG
\$FF3A	-198	BELL	MONITOR S/R TO PRINT BELL CLOSERS ACC-Y-REG
\$FF3A	-198	BELL	CALL HERE TO OUTPUT BELL
\$FF3F	-193	RESTORE	MONITOR & SWEET-16 MEMORY LOCATION 'RESTORE'
\$FF44	-188	RESTRI	MONITOR MEMORY LOCATION 'RESTRI'
\$FF4A	-182	SAVE	MONITOR & SWEET-16 MEMORY LOCATION 'SAVE'
\$FF4C	-180	SAVI	MONITOR MEMORY LOCATION 'SAVI'
\$FF59	-167	RESET	CALL HERE HAS SAME EFFECT AS PUSHING RESET BUTTON
\$FF63	-153	MON	MONITOR S/R - NORMAL ENTRY TO 'TOP' OF MONITOR WHEN RUNNING
\$FF69	-151	MONZ	MONITOR S/R TO RESET AND ENTER MONITOR
\$FF73	-141	MONZ	MONITOR MEMORY LOCATION 'MONZ'
\$FF7A	-134	CHSRCH	MONITOR MEMORY LOCATION 'CHSRCH'
\$FF7C	-132	ZMODE	MONITOR & MINIASSEMBLER MEMORY LOCATION 'ZMODE'
\$FFBA	-118	DIC	MONITOR MEMORY LOCATION 'DIC'
\$FF90	-112	NXTBIT	MONITOR MEMORY LOCATION 'NXTBIT'
\$FF9B	-104	NXTDAS	MONITOR MEMORY LOCATION 'NXTDAS'
\$FFA2	-94	NXTBSZ	MONITOR MEMORY LOCATION 'NXTBSZ'
\$FFA7	-89	GETNUM	MONITOR & MINIASSEMBLER MEMORY LOCATION 'GETNUM'
\$FFAD	-83	NXTCHR	MONITOR MEMORY LOCATION 'NXTCHR'
\$FFBE	-44	TOSUB	MONITOR & MINIASSEMBLER MEMORY LOCATION 'TOSUB'
\$FFC7	-37	ZMODE	MONITOR MEMORY LOCATION 'ZMODE'
\$FFC8	-32	CHRTBL	MONITOR & MINIASSEMBLER MEMORY LOCATION 'CHRTBL'
\$FFE3	-29	SUBTBL	MONITOR MEMORY LOCATION 'SUBTBL'
*1979/06/29 VERSION			PREPARED BY PROF H. F. LUEBBERT DARTMOUTH COLLEGE* HANOVER N H

22

Reprinted from the Cider Press of the San Francisco Apple Core Pg 10, Vol 2, No 4, August 1979

HEXLOC	DECLOC	NAME	USE
\$0000-03FF		ZERO PAGE UTILITIES	
\$0400-07FF		TEXT OR LO-RES 1	
\$0800-0BFF		TEXT OR LO-RES 2	
\$0C00-0FFF			
\$1000-13FF			
\$1400-17FF			
\$1800-1BFF			
\$1C00-1FFF			
\$2000-23FF			
\$2400-27FF			
\$2800-2BFF			
\$2C00-2FFF			
\$3000-33FF			
\$3400-37FF			
\$3800-3BFF			
\$3C00-3FFF			
\$4000-43FF			
\$4400-47FF			
\$4800-4BFF			
\$4C00-4FFF			
\$5000-53FF			
\$5400-57FF			
\$5800-5BFF			
\$5C00-5FFF			
\$6000-63FF			
\$6400-67FF			
\$6800-6BFF			
\$6C00-6FFF			
\$7000-73FF			
\$7400-77FF			
\$7800-7BFF			
\$7C00-7FFF			
\$8000-83FF			
\$8400-87FF			
\$8800-8BFF			
\$8C00-8FFF			
\$9000-93FF			
\$9400-97FF			
\$9800-9BFF			
\$9C00-9FFF			
\$A000-A3FF			
\$A400-A7FF			
\$A800-ABFF			
\$AC00-AFFF			
\$B000-B3FF			
\$B400-B7FF			
\$B800-BBFF			
\$BC00-BFFF			
\$C000-C3FF			
\$C400-C7FF			
\$C800-CBFF			
\$CC00-CFFF			
\$D000-D3FF			
\$D400-D7FF			
\$D800-DBFF			
\$DC00-DFFF			
\$E000-E3FF			
\$E400-E7FF			
\$E800-EBFF			
\$EC00-EFFF			
\$F000-F3FF			
\$F400-F7FF			
\$F800-FBFF			
\$FC00-FFFF			

MEM

H I M E H

# Graphics Driver for the IDS 440 Printer

Hersh Pilloff

The Washington Apple Pi newsletter of November 1979 provided details for interfacing the IDS 440 "Paper Tiger" printer via the Game I/O together with a software driver for printing text. The following information contains the necessary modifications for adapting the graphics driver, written by Ron & Darrell Aldrich for the IDS 225 and published in the April-May 1979 Call-A.P.P.L.E., to the IDS 440.

A call to Don Williams quickly revealed the two critical patches- In the machine language program change 0C48 : 03 to 0C48 : 05 and in the Integer BASIC routine, line 220, change STEP 4 to STEP 6. In addition, line 150 should be deleted. Note that the CALL 935 in line 210 sets the horizontal dot spacing appropriate to 16.5 characters per inch whereas the printer manual specifically cautions against using this, the highest dot density, for plotting because of potential damage to the print head resulting from thermal overload as well as the possibility of plotting errors because the dot printing frequency may exceed the timing constraints of the microprocessor. The easiest way to avoid this is to delete the CALL 935 in line 210 and check that the printer has been set for a lower print density.

An alternative approach, compatible with the AP.EDIT Program Line Editor which intercepts some of the same control characters from the keyboard used by the printer, is to modify the text printer driver to include CALLS for all possible print densities as well as enhanced characters. A listing of this driver "PRINTER" (300.300) is given here together with the appropriate CALLS. Note that "PRINTER" has been set to print 80 characters/line. For the readers' convenience, the modified integer BASIC routine "HIRES DUMP" and the machine language listing "HIRES BDUMP" are also given here.

p.s. One way of avoiding a syntax error when turning PRINTER off is to first hit reset, then 300G, and finally PR#0.

## PRINTER

```

0000- 98 48 A4 36 20 99 03 EA
0003- A9 50 8D AC 03 A5 24 8D
0010- F8 07 68 C0 F0 F0 04 A0
0018- 4C EA 03 A8 60 48 48 A5
0020- 21 8D AD 03 AD AC 03 85
0028- 21 2C 65 C0 10 F8 AD F8
0030- 07 05 24 68 B0 03 48 A9
0038- 40 2C 6F 03 F0 03 EE F8
0040- 07 20 70 03 68 48 90 E1
0048- 49 0D 0A 00 0D 8D F8 07
0050- A9 9A 20 70 03 A9 58 20
0058- A8 FC AD F8 07 F0 00 E5
0060- 21 E9 F7 90 04 69 1F 85
0068- 24 AD AD 03 85 21 68 60
0070- 9C 78 07 08 A0 0B 18 48
0078- 30 05 AD 59 C0 90 03 AD
0080- 58 C0 A9 14 48 A9 20 4A
0090- 93 FD 68 E9 01 D0 F5 68
0098- 6A 82 90 E3 AC 78 07 28
0100- 60 A9 1D 85 36 A9 03 85
010A- 37 50 A7 9C 4C ED FD A9
0110- 9D 4C ED FD 50 28 4C ED
0118- FD A9 9F 4C ED FD A9 81
0120- 4C ED FD 50 19 00 FF FF
0128- FF

```

```

CALL 730: (3A2) = 8.5 CFI
CALL 730: (3A7) = 10 CFI
CALL 940: (3AD) = 12 CFI
CALL 945: (3B1) = 16.5 CFI
CALL 950: (3B6) = ENHANCED MODE

```

## HIRES DUMP

```

LIST
5 REM SLIGHTLY MODIFIED FOR IDS
440 PRINTER BY H. PILLOFF
WITH THANKS TO DON WILL-
IAHS
10 REM HI-RES SCREEN DUMP ROUTINE
20 REM FOR INTEGRAL DATA
30 REM IP - 225 PRINTER

40 REM BY RON & DARRELL ALDRICH

*100 D$="":C$="":D$="":K$=""
* REM
CTRL B, C, D & K

110 X0=Y0=COLR: DIM A$(40)
120 PRINT D$;"NOMON C,I,O"
130 TEXT : CALL -936: POKE -16297
,0
140 PRINT D$;"BLOAD PRINTER": PRINT
D$;"BLOAD HI-RES BDUMP"
170 CALL -936: INPUT "NAME OF PICTUR
E ",A$: IF A$="" THEN 170
180 PRINT D$;"BLOAD ",A$;",U,A$2000"
: GR
190 CALL -936: INPUT "INVERT IMAGE T
O PRINTER ?",A$: IF A$="" THEN
190: IF ASC(A$)=217 THEN 210
: IF ASC(A$)#206 THEN 190
200 CALL 3266
210 CALL 768: REM CALL 0940: PRINT
: PRINT C$: REM CALL PRINTER: C
ALL PRINT SIZE: SET GRAPHICS

220 FOR Y0=0 TO 168 STEP 6: POKE
3302,Y0: CALL 3072: REM PRINT
LINE
230 PRINT C$:K$: NEXT Y0
240 PRINT C$:B$: PRINT D$;"PR#"

250 END

```

## HI-RES BDUMP

```

0C00- A9 20 8D F1 0C A9 00 9D
0C08- E7 0C 8D E8 0C A9 00 8D
0C10- EA 0C AD E6 0C 8D E9 0C
0C18- AD E9 0C AE E7 0C AC E8
0C20- 0C 20 66 0C AD F2 0C 29
0C28- 7F 31 26 C9 00 F0 02 A9
0C30- 40 8D EB 0C AD EA 0C 4A
0C38- 6D EB 0C 8D EA 0C EE E9
0C40- 0C AD E9 0C ED E6 0C C9
0C48- 05 D0 CD AD EA 0C 20 ED
0C50- FD AD E7 0C C7 17 AD E8
0C58- 0C E9 01 EE E7 0C D0 03
0C60- EE EB 0C 90 A8 60 8D EE
0C68- 0C 8E EC 0C 8C ED 0C 48
0C70- 29 C0 85 26 4A 4A 05 26
0C78- 85 26 68 85 27 0A 0A 0A
0C80- 26 27 0A 26 27 0A 66 26
0C88- A5 27 29 1F 0D F1 0C 85
0C90- 27 8A C0 00 F0 05 A0 23
0C98- 69 04 C8 E9 07 80 FB 8C
0CA0- F0 0C AA 8D E6 08 8D F2
0CA8- 0C 98 4A AD EF 0C 8D F3
0CB0- 0C B0 01 60 0A C9 C0 10
0CB8- 08 AD F3 0C 49 7F 8D F3
0CC0- 0C 60 A9 20 8D F1 0C 85
0CC8- 27 A0 00 84 26 81 26 49
0CD0- FF 91 26 C8 D0 F7 E6 27
0CD8- A5 27 29 1F D0 EF 60 81
0CE0- 82 84 88 90 A0 C0 00 18

```





# 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

Authorized service: APPLE  
Micropolis  
Dynabyte

soon to add: Thinker Toys

---



"The Plus Makes the Difference"