HOUSTON AREA APPLE USERS GROUP

News from the Apple Barrel

Volume #2 No. 2 June, 1979

President, Dewayne Van Hoozer Editor, Ed Seeger

*** NEW APPLE PRODUCTS ***

APPLE COMPUTER INC. has announced a number of new peripherals and exhibited them this past week in Chicago at the National Computer Conference. Scheduled for release over the coming summer are such goodies as a GRAPHICS TABLET, AUTO-START ROM (to boot your DOS upon turning on the system), a CLOCK-CALENDAR CARD (from Mountain Hardware), the DISK UTILITY PACK (DOS 3.2, manual, and several utility routines), a DESK and "DESK RETURN" (actually a sort of credenza to hold a printer alongside the desk), and then yes, Virginia, Uncle Apple will soon take orders for THE LANGUAGE SYSTEM, better known as PASCAL! List price -- $495.

THE LANGUAGE SYSTEM, will be a three-language ROM card that will load Integer Basic, Floating Point Basic, or Pascal into an additional 16K of RAM. Got it? The Apple II will in effect become a 64K machine. Our Applesoft ROMcards have been a bust for those who choose to use Pascal. Apple owners who have held off on springing $200 for the Applesoft ROMcard will be in luck, however, as a used market develops for these things. Use your tape or disk versions just a little longer; your patience will be rewarded! Delivery of the new system is forecast for September. Funny how you engineering types who feared foreign languages in college can't wait to learn Pascal! "What a chimera then is man! What a novelty! What a monster, what a chaos, what a contradiction, what a prodigy! Judge of all things, feeble worm of the earth, depository of truth, a sink of uncertainty and error, the glory and the shame of the universe." -- Blaise Pascal

P R O G R A M M E R ' S   A I D   # 1

by
Dennis A. Cornwell

One of the least expensive (and thus surprisingly underpublicized) accessories for the APPLE II is the Programmer's Aid #1. The purpose of this article is to discuss its features plus give some of this author's highly subjective views on its value.

The PA#1 is a single 2K-byte ROM chip which plugs into socket D0 of your APPLE. It contains a "library" of routines which integer Basic users frequently need but don't always have easy access to. It really is an extension of the monitor and integer Basic that comes built into your machine

The PA#1 has six main capabilities:

- program renumbering and appending
- tape verification (both Basic and machine language)
- machine language program relocation
- RAM memory testing
- music tone generation
- hi-res graphic generation

Once the chip is installed (which the accompanying manual shows you how to do), all of the above are accessable from your keyboard without additional loading, etc.

Before going into the details of each routine, a few general comments. First, the documentation is excellent. The manual is over 100 pages long and is full of examples (set in green type to stand out), has all the source assembly listings, plus a terrific summary of all PA#1's commands. This summary is in the back of the book which makes it very handy to flip to when you're using a routine and don't need all the details about it-just a refresher course, so to speak.

Second, it doesn't take away from any of your usable RAM, thereby reducing the space availible for your program. It actually will allow you to write longer programs since it eliminates a lot of coding you had to have in before. The PA#1 routines themselves reside in $D000 - $D7F8 (53248 - 55288), just under the Basic and monitor ROM areas.

----Program renumbering and appending

How many times have you written a program and then wanted to insert a new block of code--only to find that you don't have sufficient line numbers in the desired area? Or wanted to tack one routine on the end of another? Well, this sure beats doing it through the keyboard one line at a time.

This section is really three (3) separate routines:

- renumbering an entire program
- renumbering only a portion within a program
- adding one program to another

Let's say you have a program that was numbered like this:

```
    7
  213
  527
  698
13000
13233
```

(must have been done late at night!)
and wanted a little more 'continuity' in the numbering. With the PA#1, you would type

```
CLR
START = 1000
STEP = 10
CALL -10531
```

The following would appear on the screen:

```
 7 -> 1000
 213 -> 1010
527 -> 1020
698 -> 1030
13000 -> 1040
13233 -> 1050
```

Now, not only are your lines renumbered but also any references to a line number are updated (such as a GOTO, GOSUB). However, if an arithmetic expression or variable is used in a GOTO or GOSUB, that expression will generally not be renumbered correctly. For example, GOTO TEST or GOSUB 10 + 20 will not be renumbered. (This might be something that a later version could at least flag for you--are you listening, APPLE?)

Now, let's say you have a program that was numbered like this:

```
 100
 120
 300
 310
 402
 500
2000
2022
```

and you want to renumber only the lines between 300 and 500. You would type:

```
CLR
START = 200
STEP = 20
FROM = 300
TO = 500
CALL -10521
```

The following would appear on the screen:

```
300 -> 200
310 -> 220
402 -> 240
500 -> 260
```

and the new program line numbers will be:.

```
100
120
200
220
240
260
2000
2022
```

There is one catch to this--you can change line 'numbers' but not the line 'order'. In other words, if you wanted to insert lines 2000 and 2022 between lines 120 and 200, you couldn't renumber them to 150 and 160, respectively. (Another suggestive hint from the users, APPLE!)

Note: Didja ever wonder how people put line numbers in their programs that were higher than 32767? Well, this section is one way--try renumbering using a 'negative' STEP value. Gives some interesting results.

To combine two programs (or portions of programs), you first need to renumber them so that there is no overlap on line numbers (i.e., one program's number must be greater than the other's). Let's call the program with the high numbers, Program 2, and the one with low numbers, Program 1. (pretty high level stuff, eh?) Now load Program 2 into the APPLE using the normal LOAD command. Then put the Program 1 tape into the recorder, type CALL -11076 and start the recorder again. This will give the usual two beeps (if loaded successfully) and your two programs are now linked together (assuming there's sufficient memory for both).

---Tape verification (Basic and machine language)

If you're like I am, you've occasionally spent several minutes (and a few droplets of perspiration) wondering whether your only version of a program got saved correctly to tape. And to further add to the dilemna, the only way to check is to re-load the just taped version--thus destroying what previously was in memoery. So if the SAVE wasn't good, you just destroyed what was.

The PA#1 eliminates this worry by comparing what's on the tape and what's in memory. Any changes between the two versions prompts an error message. You can then go back and re-SAVE the tape.

To verify integer Basic, save your program in the usual way, rewind the tape and type:

```
CALL -10955
```

If the APPLE beeps its usual two times, the tape is okay. Otherwise, the ERR message occurs.

To verify a machine language write, first write the desired portion of memory as usual:

```
START.STOPW (where STOP & START are in hex)
(rewind tape)
D52EG
START.STOP cntl Y
```

and start the tape again. Any difference will generate an error message and display the location involved.

Although this could be a handy feature, I seldom use it because I haven't had that much trouble with tape saves (or could it be because I now have a disk?)

---Machine language program relocation

Although I run the risk of offending some of H.A.A.U.G.'s more technical-minded members, this feature is one I've never used. Because I never used it (or machine language to any large degree), this section will be more on 'how' to use the feature it than on 'what' it might be used for.

In simplest terms, these routines take blocks or segments of machine code and/or data and modifiies them so that they run correctly in a different location of memory.

It says it can be used to combine two programs which ran from the same memory area separately but now must run concurrently; place ROM code into RAM (or vice versa) for 'experimental' purposes; or move a routine (or group of routines) to different locations.

To 'relocate' a block of 'code', type (from the monitor):

```
                    D4D5G
                    ADDRESS3<ADDRESS1.ADDRESS2 cntl Y
```

where

```
          ADDRESS3 is the destination area beginning
          ADDRESS2 is the source       "      "
          ADDRESS1 is the   "          "    ending
```

To 'move' a block of 'data', type

```
                    D4D5G
                    ADDRESS3<ADDRESS2.ADDRESS1 M
```

Notice that code is handled differently from data due to the need to handle pointers, multi-byte attributes of code, etc.

Here again, I will leave it to some of our more 'gut-level' members to fill the above gap.

---RAM memory testing

For those of you have bought (or plan to buy) additional RAM capacity, this is a reassuring little feature to have at the time of purchase and even beyond. The testing is done by writing a number to each memory address in a specified test range, and then reading the number stored at each address, and comparing.

Two types of errors can be detected: simple and dynamic. A 'simple' error occurs when the number written to an address is not the same number when read back from the 'same' address. A 'dynamic' error occurs when the writing of a number to one address causes a number to change at a 'different' address.

To run a test, hit 'reset' and then type:

```
                    D5BCG
                    a . p cntl Y
```

where

'a' is the starting address of the test range, and
'p' is the number of 256 byte (100 hex) 'pages' you wish to test. 'p' * 100 cannot be > 'a'

A complete test for a 48K system is:

```
                 D5BCG
                 400.4 cntl Y     Test for Row 'C' (screen)
                 800.8   "                   "
                 1000.10 "                   "
                 2000.20 "                   "
                 3000.20 "        Boundary test between C/D
                 4000.40          Test for Row 'D'
                 7000.20          Boundary test between D/E
                 8000.40          Test for Row 'E'
```

The above can be typed separately or all on one line as a single command. It takes about 3 - 4 minutes to run a 48K test.

In addition to this single cycle test, you can also do an automatic repeating test. This tests continously until you stop it or it detects an error. To run this, type

```
D5BCG
N (followed by the desired tests) 34:0
```

Although this is a useful test, it's dull to watch except when doing the screen area when it produces an interesting graphic effect.

I am also trying to figure a way to store the test instructions in a disk file rather than having to retype it each time. Anyone figure this one out yet?

---Music tone generation

For those of us that enjoy the musical aspects of the APPLE, there were two (at least) ways to produce sounds. One is the rather obtuse method of PEEKing and POKing location -16336. The other was using the machine language routine described in the red APPLE II Reference Manual. Both are somewhat limited.

The PA#1 music routines go a level above the former by letting the programmer control the pitch (high or low note), duration and timbre (a tone quality factor).

There are 50 notes available (1 to 50) in the routines with note # 32 about middle 'C'. Any increment of one for the pitch variable produces a semitone difference. Thus note # 33 is 'C' sharp, etc.

Duration can be be assigned a value of between 1 and 255 with 170 being equal to about one second. Timbre can have five values: 2, 8, 16, 32, and 64.

These variables (pitch, duration and timbre) are set from within the program by POKing locations 767, 766, and 765, respectively. The note is played by CALLing location -10473.

A sample program for the playing of a chromatic scale of four octaves:

```
10 MUSIC =  -10473
20 PITCH = 767: TIME = 766: TIMBRE = 765
30 POKE TIME, 40: POKE TIMBRE, 32
40 FOR I = 1 TO 49
50 POKE PITCH, I : CALL MUSIC
60 NEXT I : END
```

----Hi-res graphic generation

The largest section in the PA#1 manual is dedicated to hi-res graphic routines. It is an extensive and interesting section which, once mastered, can really help you produce a great deal of slick output with considerable ease.

One of the biggest gripes I originally had with my APPLE was the awk-wardness of using hi-res from integer Basic. First you had to load the machine subroutines from tape. Then you had to POKE yourself til you were blue in the finger generating the desired display. When I finally got my APPLESOFT/ROM-card, there was a quantum leap in programming ease--no tapes plus a greatly reduced requirement for POKE's.

The PA#1 gives the integer Basic programmer essentially the same capabilities as does the ROM version of APPLESOFT (please make sure you're differentiating between the cassette RAM version and the ROM plug-in card). I say essentially because the PA#1 has most (but not all) of the features of the ROM card plus a few that the ROM card doesn't have. The following table illustrates both's capabilities:

Capabilities	PA#1	APPLESOFT/ROM
Initialize and clear screen	yes	yes
Up to 8 colors (only 4 on systems with s/n < 6000)	yes	yes
Solid background in a single command	yes	no
Plot or position a point (including x,y co-ords) in a single command	no	yes
Draw a single line in a single command	no	yes
Draw mutiple lines in a single command	no	yes
Display two different pages of memory	yes	yes
Display 280 x 160 matrix plus 4 lines text	yes	yes
Display 280 x 192 matrix with no text	yes	yes
Handles shape tables (DRAW, SCALE, ROT, SHLOAD)	yes	yes
Single command shape erase (XDRAW)	no	yes
Single command shape linkage (DRAW1)	yes	no
Finding a shape's last point location/color	yes	no
Determine whether two shapes collide or overlap	yes	no

As you can see above, the big drawback (pardon the pun) to the PA#1 is that it cannot handle a POSN, PLOT or LINE in a single command (assuming the color has already been specified). As an example, the following are the statements necessary to plot a white square box on a solid green screen in both integer Basic with PA#1 and APPLESOFT/ROM:

```
-------------PA#1---------------         --------APPLESOFT/ROM---------

CALL -12288: REM INITIALIZE/             HGR: REM INIT/CLEAR
CALL -12274: REM CLEAR
XO = 0: YO = 0
COLR = 42: REM COLOR = GREEN             COLOR = 1: REM GREEN
CALL -11471: REM SOLID BACKGROUND        FOR Y = 0 TO 159
                                         FOR X = 0 TO 279
                                         HPLOT X,Y : REM BACKGROUND
                                         NEXT X: NEXT Y
COLR = 255: REM COLOR = WHITE            COLOR = 3: REM COLOR = WHITE
CALL -11506: REM PLOT POINT 1
XO = 0: YO = 159
CALL -11500: REM DRAW LINE 1
XO = 279: YO = 159
CALL -11500: REM DRAW LINE 2
XO = 279 : YO = 0                        HPLOT 0,0 TO  0,159
CALL -11500: REM DRAW LINE 3                       TO  279,159
XO = 0: YO = 0                                     TO  279,0
CALL -11500: REM DRAW LINE 4                       TO  0,0: REM DRAW BOX
```

However, when comparing the PA#1 to 'normal' integer Basic, it still can save considerable time even when POSN, PLOT or LINE are used.

The PA#1 hi-res routines work under the assumption that the CALL parameters used in a program (XO,YO,COLR,SHAPE,ROT and SCALE) are the first variables defined and exactly in this order. This is because the routines, when called, look to the first locations in the APPLE's variable table for the required values. Not at the names, but at the values-hence the order.

The manual also goes into quite a bit of detail about the creating, storing and use of shape tables. These can be very useful but incredibly dull to produce and input. (the easiest way to do a table I've found is to beg, borrow or steal (buy?) a copy of Creative Computing-July/August, 1978 which has a routine to do it automatically while you just draw the shape on the screen.) The manual also goes into detail about using shape tables with a disk which is nice.

---Summary---

Overall, my feelings towards the PA#1 are very positive. It's quite versatile and very handy to use being in ROM. I think I've gotten my money's worth out of it already and I haven't used it as much as I plan to in the future.

Would I recommend that every APPLE owner rush out and buy one? Well, maybe. It depends on your situation--primarily on whether you do or do not have two things already:

- an APPLESOFT/ROM card
- a DISK II

If you have the ROM card and are a die-hard, 'floating-point phreaque', don't buy one. You have the hi-res capabilities already and you probably won't use the other integer Basic routines anyway. Save the $50 and blow $7.50 of it on a RAM test cassette.

If you have a DISK II, the decision is not as clear cut. Most of the routines on the PA#1 are available on disk (by hook or crook). And once on a disk, they're just about as accessable as being in ROM. Besides, some of PA#1's routines don't work on disk files e.g., append and verify (APPLE hint, hint!).

However, if you don't have a ROM card or a DISK II, it's well worth the $50. Just adding the estimated retail value of the separate routines could get you past the cost in a hurry. Plus it sure makes some things a heck of a lot easier to do--and the easier things are, the better I like 'em!!

**** A NOTE OF ACKNOWLEDGEMENT ****

All of the above examples came from the PA#1 "Installation and Operating Manual" plus just darn well all of the informa tion besides. I appreciate their permission to use it in the article.

DAC

*** SEE-LOAD ***

Radio Shack announces at last the solution to the heartbreak of keybounce on its TRS-80. Actually, several solutions are available. Their May, 1979, "Microcomputer Newsletter" advises that microsurgery with a paperclip may do it. Or you can buy a whole program (!) if you wish. And rumor has it that a new upper-level version of their BASIC has ended bbbbounce for ever. One Fort Worth worthy was seen last Saturday morning over in lingerie at Nieman's, trying out foundations. ("But she told me it would eliminate all my jiggle...") Probably thinks the Z-80 is a Datsun, too!

Speaking of the competition, following is a release hot off the Texas International wire:

```
T    NEWS.EOO.CHAPTER.SECTION.PAGE4
*
HOME COMPUTER ANNOUNCED.-TI JUNE 1 ANNOUNCED ITS HOME COMPUTER, MODEL TI-99/4,
TO BE INTRODUCED AT THE SUMMER CONSUMER ELECTRONICS SHOW IN CHICAGO JUNE 3-6.
  THE HOME COMPUTER SYSTEM CONSISTS OF A CONSOLE WITH 16K RANDOM-ACCESS MEMORY,
SOUND, GRAPHICS, EXTENDED 'BASIC' COMPUTER LANGUAGE, AND A 13-INCH COLOR TV
MONITOR.  IT IS TO BE AVAILABLE IN LATE SUMMER AT A SUGGESTED RETAIL PRICE OF
$1,150.  SOLID-STATE SOFTWARE COMMAND MODULES FOR THE SYSTEM WILL CARRY SUG-
GESTED RETAIL PRICES RANGING FROM $19.95 TO $69.95.
  THE COMMAND MODULES WILL ALLOW USERS TO ACCESS PROGRAMS WITHOUT NEEDING TO
BE FAMILIAR WITH COMPUTERS OR COMPUTER PROGRAMMING.  MODULES AVAILABLE AT THE
TIME OF INTRODUCTION OR BY YEAR-END INCLUDE DEMONSTRATION AND DIAGNOSTIC MOD-
ULES, AND MODULES FOR HOME FINANCIAL DECISIONS, EARLY LEARNING FUN, BEGINNING
GRAMMAR, NUMBER MAGIC, VIDEO GRAPHS, HOUSEHOLD BUDGET MANAGEMENT, VIDEO CHESS,
FOOTBALL, PHYSICAL FITNESS, SPEECH CONSTRUCTION, INVESTMENT ANALYSIS, TAX AND
INVESTMENT RECORD KEEPING, STATISTICS, AND EARLY READING.
*
GERMANIUM PRODUCTION PHASED OUT.-TI HAS PHASED OUT OF THE GERMANIUM TRANSISTOR
BUSINESS AFTER 25 YEARS AND SOME 2 BILLION UNITS OF PRODUCTION.  HISTORY OF TI
INVOLVEMENT IN THE BUSINESS WAS RECITED MAY 31 AT A LUNCHEON FOR KEY PRODUCTION
OPERATIONS AND MANAGERS ON THE PROGRAM.  TI CHMN. MARK SHEPHERD, JR., PROJECT
ENGR. FOR THE PRODUCTION START-UP IN 1953, CREDITED SUCCESS OF THE PROGRAM TO
THE TIERS INVOLVED IN IT OVER THE YEARS.//END T NEWS JUNE 1, 1979//
```

*** APPLESOFT SYMBOL TABLE PRINTER PROGRAM ***

PHIL ROYBAL, Product Marketing Manager of Apple Computer Inc., has sent along a brief sub-routine to be appended to your Applesoft programs. To quote from his accompanying letter:

"Enclosed you will find a listing of a program that will print the symbol table of any Applesoft program. This little program may be appended on the end of an Applesoft program and called as a subroutine. Of course it cannot be "RUN" since Run clears the variable table. Therefore it must be called from within a program or used with a "GOTO". In any case you will find this program handy for working over software you get from other people. It's especially useful when you're trying to find an unused variable that you can use for some sort of a patch. I hope you enjoy it."

Your editor appended it onto Swords and Sorcery!, a Middle-Earth-type adventure program, written in Applesoft, and loaded with variables. It's on vol. #14 of the HAAUG Software Library. Following is a print-out of the symbol table, along with a listing of Phil's program. Thanks, Phil, for helping us out with a program when the Houston crowd ran dry!

```
GOTO 32000

APPLESOFT SIMPLE VARIABLES


W2
N$
I3
I1
X1
V
I1$
F1
W1
C1
Y1
V1
C2
L1
T1
C3
Z1
S1
L2
L0
D0
I2
E1
P3
D3
I
P7
D4
F5

APPLESOFT ARRAY VARIABLES


H=$

)
```

```
LIST32000,32767


32000  REM      APPLESOFT SYMBOL
TABLE PRINTER PROGRAM
32001  REM      COPYRIGHT BY PHIL
 ROYBAL, 2/9/79




32002  REM  =====================

32005  HOME : PRINT "APPLESOFT SI
MPLE VARIABLES"
32007  PRINT : PRINT
32008 ZV = 0:ZW = 0:ZX = 0:ZY = 0
:ZZ = 0
32010 ZY =  PEEK (107) + 256 *  PEEK
(108)
32015 ZZ =  PEEK (105) + 256 *  PEEK
(106)
32017  IF ZZ >  = ZY - 35 THEN 32
210
32020  FOR ZX = ZZ TO ZY - 36 STEP
7
32030  GOSUB 32530
32200  NEXT ZX
32210  PRINT : PRINT "APPLESOFT A
RRAY VARIABLES": PRINT : PRINT

32220 ZZ =  PEEK (109) + 256 *  PEEK
(110)
32230 ZX =  PEEK (107) + 256 *  PEEK
(108)
32235  IF ZX > ZZ THEN 32270
32240  GOSUB 32530
32250 ZX = ZX +  PEEK (ZX + 2) +
256 *  PEEK (ZX + 3)
32260  GOTO 32235
32270  END
32530 ZV =  PEEK (ZX):ZW =  PEEK
(ZX + 1)
32540  PRINT  CHR$ (ZV); CHR$ (ZW
);
32550  IF ZV < 127 AND ZW < 127 THEN
 PRINT : RETURN
32560  IF ZV > 127 THEN  PRINT "%
      ": RETURN
32570  PRINT "$": RETURN
```

Attention users of disk S-C ASSEMBLER II !

Larry Shurr

Have you run home with your new copy of APPLE DOS 3.2, booted it up and tried to run your assembler yet? If you have, you know that it all comes crashing down around your ears. As you survey the wreckage, however, do not despair! The required patches are quite simple. Use the following procedure to create S-C ASSSEMBLER 3.2 and all your problems are solved:

```
BLOAD the assembler
CALL -151                                          (call monitor)
11D0:B
11D5:51 A8
3D0G                                               (re-enter BASIC)
BSAVE S-C ASSEMBLER 3.2,A$1000,L$C00
```

The routine this will change begins at $11C1 if you would like to look at it. It is part of the assembler's initialization to link into the DOS I/O hooks and restart addresses. You will see that the routine computes an entry point into DOS, modifies itself (shame! shame!), and jumps to that address. Of course, that entry point was changed in DOS 3.2 just to mess us up. The patches make the routine compute the correct address. Naturally, this means that S-C ASSEMBLER 3.2 will only run under DOS 3.2.

—A. B. B. S.—

```
FUNCTION :?r
MSG# TO RETREIVE (1/2) ?1
MSG # 1
SUBJ.: HAAUG ABBS
TO: ALL
FROM: DEWAYNE VAN HOOZER
DATE:

WELL IT IS FINALLY HERE. THE LONG AWAIT
AWAITED ABBS SYSTEM IS UP AND RUNNING.

THIS SYSTEM MAY BE USED FOR MANY
DIFFERENT PURPOSES. THE USEFULLNESS
OF THE SYSTEM IS COMPLETELY UP TO YOU.

ED SEGER SAID THAT YOU MAY ENTER
QUESTIONS FOR DOCTOR APPLE VIA THE ABBS
BY REPLYING DOC APPLE TO THE 'TO'
FIELD IN YOUR MESSAGES.

I HAVE ALSO BEEN INFORMED THAT ~~THE~~
MAD BOMBER WILL AT TIMES RESPON~~ED~~
TO QUESTIONS THROUGH THE ABBS.
  (SEE MESSAGE TWO FOR MORE)
```

```
MSG# TO RETREIVE (1/2) ?2
MSG # 2
SUBJ.: HAAUG ABBS
TO: ALL
FROM: DEWAYNE VAN HOOZER
DATE:

IF YOU HAVE ANY QUESTIONS OF COMMENTS
CONCERNING THE OPERATION AND/OR
PROJECTS OF THE HOUSTON AREA APPLE
USER'S GROUP, PLEASE ADDRESS THE
MESSAGES TO 'HAAUG'.

COMMENTS CONCERNING THINGS YOU WOULD
LIKE FOR HAAUG TO DO AS AN ORGANIZATION
WILL BE DISCUSSED BY THE EXECUTIVE
OFFICERS AND PRESENTED AT THE NEXT
MEETING.

IF I CAN BE OF ANY HELP TO YOU PLEASE
CALL ME AT (713) 682-2126.
          THANKS FOR YOUR SUPPORT
                 D.V.

MSG# TO RETREIVE (1/2) ?
FUNCTION :?
```

FOLLOWING IS A BRIEF LIST AND DESCRIPTION OF THE ABBS COMMANDS AND THEIR USAGE:

CTRL E--RETYPES CURRENT LINE UP TO PRESENT POSITION AND ALLOWS YOU TO CONTINUE FROM THAT POINT.

CTRL H (BACKSPACE)--ALLOWS YOU TO BACKSPACE ONE CHARACTER AT A TIME AND PRINTS A '¢' FOLLOWED BY THE CHARACTER YOU ARE BACKSPACING OVER. THIS IS THE SAME ROUTINE AS IS USED FOR DELETE OR RUBOUT INSTEAD OF TRUE DELETE. (FOR THE BENEFIT OF PRINTERS)

CTRL U (FORWARD ARROW)--STARTS YOU BACK AT THE BEGINNING OF THE CURRENT LINE BEING TYPED. (I.E. START OVER)

```
<C/R> TO CONTINUE, <E> TO END ?
```

A--APPLE 40 COLUMN. NORMALLY YOU WOULD BE ALLOWED 64 CHARACTERS PER LINE. A BELL WILL SOUND AT 59 AND ON UP TO 64 COLUMNS AT WHICH POINT YOU WOULD BE FORCED ONTO THE NEXT LINE OF TEXT. IN THE APPLE 40 MODE, THE BELL WILL RING AT 35, THEN AGAIN AT 38 AND 39. DROPPING YOU TO THE NEXT LINE AT 39. 39 WAS USED INSTEAD OF 40 TO AVOID AN EXTRA BLANK LINE BECAUSE OF THE 40 TH CHARACTER.

B--PRINT BULLETIN. PRINTS BULLETINS AT BEGINNING OF PROGRAM.

```
<C/R> TO CONTINUE, <E> TO END ?
```

D--DUPLEX SWITCH. ALTERNATELY SELECTS FULL OR HALF DUPLEX OPERATION AND INFORMS YOU OF CURRENT STATUS.

E--ENTER MESSAGE. ALLOWS YOU TO ENTER A MESSAGE INTO SYSTEM. ENTER COMMANDS ARE BASICALLY SELF EXPLANATORY. A CARRIAGE RETURN (C/R) AT THIS POINT WILL LIST OUT THE COMMAND MENU FOR ENTRIES. THE CHANGE COMMAND ALLOWS YOU TO CHANGE AN ENTIRE LINE BUT NOT JUST CHANGE PART OF IT. MAKE SURE WH YOU ARE DONE WITH THE MESSAGE TO SAVE IT TO DISC WITH THE 'S' COMMAND.

```
<C/R> TO CONTINUE, <E> TO END ?
```

```
INPUTLINE >4 CHARACTERS
REENTER
?
```

G--GOODBYE. EXIT PROGRAM.

H--HELP. PRINTS THIS ROUTINE.

K--KILL A MESSAGE. ENTER THIS TO DELETE A MESSAGE FROM THE FILE. A PASSWORD MAY BE NECESSARY IF ONE WAS USED AT THE TIME OF MESSAGE ENTRY.

L--LINE FEED ON/OFF. NORMALLY ON. FOR TERMINALS THAT NEED AN EXTRA LINEFEED CHARACTER TO ADVANCE TO THE NEXT LINE.

```
<C/R> TO CONTINUE, <E> TO END ?
```

N--NULLS. ADDS AN EXTRA DELAY AFTER A CARRIAGE RETURN TO ALLOW PRINTERS TIME TO MOVE THE PRINTERHEAD BACK TO STARTING POSITION. THIS OPTION ONLY WORKS WITH THE LINE FEED OPTION ON. EACH NULL IS EQUIVALENT TO 30 MILLISECONDS DELAY AND IS ADJUSTABLE FROM 1 TO 30. IT DEFAULTS TO ONE.

Q--QUICK SCAN. AN ABBREVIATED SCAN. SEE 'S'

R--RETRIEVE MESSAGES. ALLOWS YOU TO RETRIEVE A MESSAGE FROM THE FILE.

```
<C/R> TO CONTINUE, <E> TO END ?
```

S--SUMMARIZE MESSAGES. ALLOWS YOU TO SCAN OVER MESSAGES STARTING AT THE MESSAGE # YOU SPECIFY.

T--TIME AND DATE. GIVES YOU THE CURRENT TIME AND DATE. THIS IS ALSO USED AUTOMATICALLY DURING LOG-IN.

W--WELCOME. PRINTS WELCOME MESSAGE AT BEGINNING OF PROGRAM.

X--EXPERT USER. DOES AWAY WITH CERTAIN EXPLANATORY MESSAGES DURING THE PROGRAM. IT ALSO ALLOWS CERTAIN C/R DEFAULTS. EX: A C/R IN RESPONSE TO FUNCTIONS? WILL PRINT FUNCTIONS SUPPORTED BY THE SYSTEM.

<CR> --PRINTS FUNCTIONS SUPPORTED IN THAT CURRENT MODE OF OPERATION.

*** ASK DR. APPLE ***

+++ IS THERE A WAY OF PREVENTING THE ACCIDENTAL PRESSING OF THE RESET KEY?

1) Yes. Push the button only on purpose.
2) In the Hardcopy Library there is a monograph on how to relocate the reset key to somewhere else.
3) You could put a hinged plastic case over it. Most hobby stores have small plastic hinges which are normally used in building model airplanes, but could be used to help with the reset key problem.

+++ DO YOU HAVE ANY SUGGESTIONS FOR A FILTER TO PROTECT MY APPLE FROM LINE SURGES? -- David Novak

Well, David, it is the opinion of the old Apple Doctor that you don't need one. The switching power supply in the Apple protects it from most of the garbage on the line. However, CORECOM makes two models which might be what you're looking or. Models 1EF1 and 1EF2 each sell for about $9.79. Two Houston stores sell them: Component Specialties, at 771-7237 and Newark Electronics, at 782-4800. A second product you might want to consider is produced by Electronic Specialists, Inc., Box 122, Natick, MA 01760. Their device is a black box which has 6 protected 3-prong sockets. This box sells for $49.95. See their ad on page 125 of the June issue of Interface Age magazine.

+++ WHAT IS A GOOD CHARACTER GENERATOR? -- Marshall Martin

1) They say the Army builds men. I wonder if they're LSI or VLSI?
2) There are some pretty strong characters coming out of Rice University.
3) Hardware-wise, there's the lower-case chip by Dan Paymar which goes for $49.95. Then there's the Super-Rip (uh, super-CHIP), which will be written-up in the next Apple Barrel. Then there are three software approaches. Take the excellent Screen Machine by Programma for $15.95. Also coming out is Apple's own hi-res character generator, to be isued as part of a users' Contributed Software Bank. You may find a copy floating around the club. Then there's a "Font-Set" advertised for $19.95 in the April-June 1979 Programmers Software Exchange catalog. Ed Seeger has seen a lot of this stuff. Apple Barrel hopes to review both the chip and the several programs in an up-coming issue.

+++ DO YOU KNOW OF A GOOD FLOATING-POINT PACKAGE? M. Martin

1) Never seen a point.

. ↑ not a point

2) How do points float, face up or face down?
3) Do points have hair?
4) I believe there was a pretty complete package detailed in "Dr. Dobbs" March & April issues. There's nothing on the market that provides a complete floating point package. We are starting to see some hardware boards built around calculator chips coming available. No information yet about the precision or the speed of these new beasties.

+++ WHAT IS THE PRECISION OF THE UCSD PASCAL? -- M. Martin

Larry Shurr said he would call his friends at UCSD with that question. LARRY will tell you ALL About PASCAL next month.

PREZ SEZ

by: Dewayne Van Hoozer

This month HAAUG will be trying several new ideas. First in addition to our regural Wednesday night meeting on June 13th we will also have a second meeting on Saturday June 30th at 1:00 pm in the Houston Amature Radio Club building. The HARC building is located at 7011 Lozier which is east of the Astrodome and several block south of Old Spannish Trail. If you have problems locating the building on Saturday please call 747-5073 for help.

At the second meeting we will have several things going on at once. We will have a software swap, beginners software seminar, and the first of our summer series of hardware construction seminars. The first device we will build is an autoboot device designed and prototyped by John Brisbin. In order for us to have all the parts ready in time you will have to register for the seminar by June 20th. The registeration fee is $40.00. This fee can be payed anytime prior to the seminar.

The other projects currently being designed are an originate/answer modem, RS-232 C interface and a realtime clock. If you would like to help in the design of the projects or if you have a project of your own that you think we might be interested in please give me a call at 682-2126.

The Apple Bulletin Board System is finally here. We do not yet have a perminate location for the ABBS. It is currently located at Computercraft. You can access the ABBS by calling 977-7019 after 6:00pm.

Speaking of modems, communications and such, Computercraft is offering to HAAUG members a discount on the D.C. HAYES Micromodem II. See Richard for details. Since I'm on the subject of stores, I'd like to welcome a new APPLE dealer to town. Interactive Computers is now selling APPLE hardware. Long noted for their extensive collection of APPLE software, Interactive now offers the complete APPLE line. I guess Bill Rogers finally SOL the light....

see ya at the meetings

Dewayne

Dewayne Van Hoozer

Dieter Muller, of Duncan Micro, 1015 Hickory Avenue, Duncan, Oklahoma 73533, is the author of this program, an earlier version of which is on HAAUG Software Library volume #35. The video display from this Binary Programmer is the closest thing to a front panel entry the Apple user is likely to see! Entering 1's and ∅'s at the machine level is intimacy plus with one's computer, but leaves no question of the value of an assembler, or a high-level interpreter such as BASIC. Try it. You'll learn in a new way both the simplicity and the sophistication of a microprocessor.

BINARY PROGRAMMER DOCUMENTATION
VERSION 2.1

THE BINARY PROGRAMMER V2.1 HAS 10 COMMANDS AND A COMPLETE BINARY ADDRESS AND DATA BUS.

THE COMMANDS ARE:
(ESC) - THIS KEY WILL DO A CALL TO THE ADDRESS SPECIFIED ON THE ADDRESS BUS.
(BKSP) - WILL CLEAR THE ADDRESS BUS.
(COPY OR CTRL-U) - WILL CLEAR THE DATA BUS.
(RETURN) - DECREMENTS THE ADDRESS BUS BY ONE.
(O) - INCREMENTS THE ADDRESS BUS BY ONE.
(P) - ACTIVATES OR DE-ACTIVATES THE WRITE PROTECT FEATURE. TO DE-ACTIVATE YOU MUST KNOW THE LOCK WORD.
(9) - ACTIVATES THE ADDRESS MODE (NEEDED BECAUSE THE SAME KEYS ARE USED FOR DATA AND ADDRESS ENTRY.)
(0) - SAME AS (9), EXCEPT ACTIVATES DATA MODE.
COMMANDS (8) AND (9) ARE MUTUALLY EXCLUSIVE, ONE DEACTIVAT THE OTHER.
(:) - WRITES WHAT IS ON DATA BUS TO ADDRESS ON ADDRESS BUS.
(-) - READS DATA FROM ADDRESS ON ADDRESS BUS AND DISPLAYS BELOW DATA BUS.

THE ENTRY KEYS ARE:
(1-8) DATA BUS AND FIRST 8 BITS OF ADDRESS BUS. THEY ARE READ ON KEYBOARD AS ON SCREEN. (1 IS FIRST BIT, 2 IS SECOND ETC.)
(Q, W, E, R, T, Y, U, I)-SECOND 8 BITS OF ADDRESS. READ LIKE FIRST 8 (THESE ARE ACTIVE IN ADDRESS MODE.)

NOTES

THE REACTION TIME ON THIS PROGRAM IS RATHER LOW. YOU MUST BE CAREFUL AND MAKE SURE IT READS YOUR KEYSTROKES. WHEN YOU RUN THE PROGRAM IT WILL COME UP WITH WRITE PROTECT ON. THE CLEAR ADDRESS AND DATA BUS COMMANDS DE-ACTIVATE BOTH AD MODE AND DATA MODE. WHILE IT IS POSSIBLE TO HAVE BOTH MODES DE-ACTIVATED, IT IS NOT POSSIBLE TO DE-ACTIVATE WRITING EXCEPT WITH THE WRITE PROTECT SWITCH. THE REASON THIS FUNCTIO IS INCLUDED IS TO PROTECT ANY ACCIDENTAL PROGRAM CHANGING. THE LONGEST IT WILL TAKE TO RESPOND IS ABOUT 5 SECONDS. THE REASON IS, IT HAS A TABLE TO LOOK THROUGH, THE APPROPRIAT SUBROUTINE CALL (WHICH INCLUDES A WHILE FOR ANY BINARY TO DECIMAL, DECIMAL TO BINARY CONVERSION, A RATHER SLOW PROCESS. AND THEN REPRINTING ALL BUT THE DATA READ LINE OF DISPLAY (IN OTHER WORDS, IT REPRINTS ALL CAPTIONS AND DATA EXCEPT FOR DATA READ.)

THIS PROGRAM IS WRITTEN IN APPLESOFT.

```
LIST:REM      [illegible]NARY- PROGRAMMER (V 2.1)

10  DIM B(26),ADDRESS(16),DTA(8),ACT(8)
20  FOR A = 1 TO 8:B(A) = 0:ADDRESS(A) = 0:DTA(A) = 0: NEXT A
30  FOR A = 9 TO 16:ADDRESS(A) = 0:B(A) = 0: NEXT A
40  FOR A = 17 TO 26:B(A) = 0: NEXT A
50  REM  KEYS AND FUNCTIONS:
60  REM  ------------------
70  REM    1-8,Q-I DATA KEYS
80  REM    9       ADDRESS SET
90  REM    0       SET UP DATA
100  REM    :       WRITE DATA
110  REM    -       PRINT BYTE VALUE
120  REM    P       WRITE PROTECT
130  REM  -------------------------
140 KD =  - 16384:KBDS =  - 16368: REM  KEYBOARD POINTERS
150 B(1) =  ASC ("1"):B(2) =  ASC ("2"):B(3) =  ASC ("3"):B(4) =  ASC ("4"):B(5) =  ASC ("5"):B(6) =  ASC ("6"):B(7) =  ASC (
     "7"):B(8) =  ASC ("8")
160 B(9) =  ASC ("Q"):B(10) =  ASC ("W"):B(11) =  ASC ("E"):B(12) =  ASC ("R"):B(13) =  ASC ("T"):B(14) =  ASC ("Y"):B(15) =
      ASC ("U"):B(16) =  ASC ("I")
170 B(17) =  ASC ("9"):B(18) =  ASC ("0"):B(19) =  ASC (":"):B(20) =  ASC ("-"):B(21) =  ASC ("P")
180 B(22) =  ASC ("O"):B(23) = 13:B(24) = 27:B(25) = 8:B(26) = 21
190  REM  SET VALUES TO BE READ
200 CLR =  - 936:INV =  - 384:NRL =  - 380: REM  POINTERS TO UTILITIES
210 ADD = 1:WRITE = 0:PROTECT = 1: REM     POINTERSFOR FUNCTION
220  CALL CLR: CALL NRL
230  VTAB 10: HTAB 18: PRINT "ADDRESS"
240  VTAB 13: HTAB 19: PRINT "DATA"
250  VTAB 2: HTAB 7: PRINT "ADDRESS";: HTAB 18: PRINT "DATA";: HTAB 28: PRINT "WRITE": HTAB 7: PRINT "MODIFY";: HTAB 17: PRINT
     "MODIFY";: HTAB 27: PRINT "PROTECT"
260  VTAB 16: HTAB 16: PRINT "DATA READ"
270  VTAB 9
280  FOR A = 1 TO 16
290  HTAB 4 + A * 2: PRINT ADDRESS(A);
300  NEXT A
310  VTAB 12
320  FOR A = 1 TO 8
330  HTAB 12 + A * 2: PRINT DTA(A);
340  NEXT A
350  CALL NRL
360  VTAB 1
370  HTAB 10
380  PRINT ADD;: HTAB 20: PRINT WRITE;: HTAB 30: PRINT PROTECT
390  VTAB 10 + (2 * WRITE)
400 X =  PEEK (KD): IF X < 128 THEN 400
405 X = X - 128
410  POKE KBDS,0
420  FOR A = 1 TO 26
430  IF X = B(A) THEN 460
440  NEXT A
```

```
450  GOTO 400
460  IF A > 16 THEN 550
470  IF ADD THEN 500
480  IF WRITE THEN 520
490  GOTO 400
500 ADDRESS(A) =  NOT ADDRESS(A)
510  GOTO 230
520  IF A > 8 THEN 230
530 DTA(A) =  NOT DTA(A)
540  GOTO 230
550  IF A = 17 THEN ADD =  NOT ADD
560  IF A = 17 AND WRITE THEN WRITE =  NOT WRITE: GOTO 230
570  IF A = 18 THEN WRITE =  NOT WRITE
580  IF A = 18 AND ADD THEN ADD =  NOT ADD: GOTO 230
590  IF A = 21 THEN 1160
600  IF A = 19 THEN 660
610  IF A = 20 THEN 900
620  IF A = 22 THEN 1180
630  IF A = 23 THEN 1250
640  IF A = 24 THEN 1330
643  IF A = 25 THEN 5000
645  IF A = 26 THEN 5050
650  GOTO 230
660  IF PROTECT THEN 1140
670  REM
680 OFFSET = 16
690 NUM = 0:VL = 0
700  FOR A = 1 TO 16
710 OFFSET = OFFSET - 1
720 NUM = NUM + ADDRESS(A) * (2 ^ OFFSET)
730  NEXT A
750 OFFSET = 7
760  FOR A = 1 TO 8
770 VL = VL + (DTA(A) * (2 ^ OFFSET))
780 OFFSET = OFFSET - 1
790  NEXT A
800  REM
820  FOR W = 1 TO 300: NEXT W
830  POKE NUM,VL
833  PRINT "DATA WRITTEN"
835  FOR W = 1 TO 500: NEXT W
840  GOTO 220
850  VTAB 23: PRINT "PROPOSED ADDRESS IS WRITE PROTECTED"
860  FOR A = 1 TO 16:ADDRESS(A) = 0: NEXT A
870 PROTECT = 1
880  FOR W = 1 TO 300: NEXT W
890  GOTO 220
900  GOSUB 980
910  VTAB 15: CALL NRL
920  FOR A = 1 TO 8
930  HTAB 12 + A * 2: PRINT ACT(A);
```

```
940  NEXT A
950  GOTO 230
960  CALL NRL
970  END
980 OFFSET = 16
990 NUM = 0:VL = 0
1000  FOR A = 1 TO 16
1010 OFFSET = OFFSET - 1
1020 NUM = NUM + ADDRESS(A) * (2 ^ OFFSET)
1030  NEXT A
1050 Z =  PEEK (NUM)
1060 OFFSET = 7
1070  FOR A = 1 TO 8
1080 ACT(A) =  INT ((Z / (2 ^ OFFSET))) = 1
1090  IF ACT(A) THEN Z = Z - 2 ^ OFFSET
1110 OFFSET = OFFSET - 1
1120  NEXT A
1130  RETURN
1140  VTAB 23: PRINT "WRITE PROTECT ON"
1150  FOR W = 1 TO 300: NEXT W: GOTO 220
1160 Z$ = "UNLOCK": IF  NOT PROTECT THEN 1167
1161  VTAB 23: PRINT "CODE?";
1163  FOR Z = 1 TO 6: GET X$: IF X$ <  >  MID$ (Z$,Z,1) THEN 1170
1165  NEXT
1167 PROTECT =  NOT PROTECT
1170  GOTO 220
1180  IF  NOT ADD THEN 230
1185  FOR A = 16 TO 1 STEP  - 1
1190  IF ADDRESS(A) = 0 THEN 1215
1200  NEXT A
1210  FOR A = 1 TO 16:ADDRESS(A) = 0: NEXT A: GOTO 1240
1215  IF A = 16 THEN 1230
1220  FOR B = A + 1 TO 16:ADDRESS(B) = 0: NEXT B
1230 ADDRESS(A) = 1
1240  GOTO 230
1250  IF  NOT ADD THEN 230
1255  FOR A = 16 TO 1 STEP  - 1
1260  IF ADDRESS(A) = 1 THEN 1290
1270  NEXT A
1280  FOR A = 1 TO 16:ADDRESS(A) = 1: NEXT A: GOTO 1320
1290  IF A > 16 THEN A = 16
1295  IF A = 16 THEN 1310
1300  FOR B = A + 1 TO 16:ADDRESS(B) = 1: NEXT B
1310 ADDRESS(A) = 0
1320  GOTO 230
1330 OFFSET = 16
1340 NUM = 0:VL = 0
1350  FOR A = 1 TO 16
1360 OFFSET = OFFSET - 1
1370 NUM = NUM + ADDRESS(A) * (2 ^ OFFSET)
1380  NEXT A
```

```
1400  CALL NUM
1410  PRINT "END OF SUBROUTINE"
1420  FOR W = 1 TO 1000: NEXT W
1430  GOTO 220
5000  FOR A = 1 TO 16:ADDRESS(A) = 0: NEXT A
5010 ADD = 0:WRITE = 0
5020  GOTO 220
5050  FOR A = 1 TO 8:DTA(A) = 0: NEXT A
5060 ADD = 0:WRITE = 0
5070  GOTO 220
```

"FIX YOUR SWEET SIXTEEN?"

The sw16 disassembler needs to skip three bytes on the entry jump. One way to fix it, also contributed by Dieter Muller, is this:

```
610 CURRENT = CURRENT + 2
620 BYTE3 = PEEK (CURRENT-1)
640 GOTO 522
```

*** WRITING FOR APPLE BARREL ***

APPLE BARREL, like HAAUG itself, is a user-oriented medium. Neither one is the creation of someone else for the benefit of the user. Each exists as a creation of users, by users, for users. The articles and programs you enjoy here are the output of other Apple hobbiests like yourself, who have worked to wrest into a communicable form something they have conceived, nurtured and given birth to. It has been observed that the ability to communicate one's idea to others is a test of one's comprehension of his or her own idea. Apple Barrel, then, is a proving ground for your growing mastery of BASIC and 6502 assembly languages, and is a worthy forum wherein to demonstrate your skills at self-understanding and communication in areas related to you and your Apple .

There are several points to consider in writing either an article or program for publication here.

1. ASSUME that your reader, although somewhat knowledgeable about the Apple, understands only some fundamentals, but wants to learn more. Your job, then, is to teach.

2. KEEP IN MIND the faithful guidelines of WHO, WHAT, WHERE, WHEN, WHY and HOW. Take time to explain. If it is a program you are sharing, tell what it does, where it loads, when it is useful, why you wrote it, and then go on to discuss some of how it works.

3. ILLUSTRATE, simply and concretely, the things you are writing about. If printing all or part of a program run will get your point across better, submit it. You can figure your readers will retain only a little of what you tell them, but will hang on to a good bit of what you can show them. Ideally, your reader should be able to try out your item on an Apple (if it is a program you have written), and you will thereby assure the greatest possible learning and retention.

YOU DON'T HAVE TO HAVE IT ALL TOGETHER yourself to write for APPLE BARREL. Each of us is at a unique and different stage of computer mastery. If you really have no idea what the sub-routine at $9E7E does, don't worry about it. Somebody else in the club DOES know, or can figure it out. Your questions are as valuable as your answers.

Typed copy is fine for submission to Apple Barrel, but there are other options as well. The editor can accept materials done on the more popular text-editors, such as "Dr. Memory," "ApplePIE," "APPLEcations Unlimited Text

Editor," or even the "Appen 1." Submit disk (or tape, in the case of Appen 1) and a blank will be returned to you. Please try to catch your errors, both syntactic and typographical, prior to submission. Your editor is a one-finger typist (different from a one-fingered typist) and cannot retype messy manuscripts. If you have or have access to a printer, run things through on your own and send in an original printout. Dot matrix copies reasonably well, but letter-quality Selectric is crispest of all.

If one of the club officers asks you to work up an article, take it seriously. You have been asked because we are aware that you are doing something interesting, and the rest of us would like to learn from you. What do you say?

Submit manuscripts to:

Ed Seeger, Editor
The Apple Barrel
4331 Nenana Drive
Houston, Texas 77035

(713) 723-6919

*** PRINTING APPLE BARREL ***

APPLE BARREL is printed solely through club dues and special donations. HAAUG thanks Rudge Allen for making available the resources of his office for duplicating the last issue of the newsletter. An occasional donation such as that will help insure issues of the size and content we think members want, while keeping within our financial limits. Do any other HAAUG's have access to high-speed, two-sided copy equipment? Please let Ed Seeger know if you are able to donate printing.

HAAUG ROSTER

ALLEN, E. RUDGE...713-622-3979 :21
ATKINSON, E. NEELY...713-529-4165 :105
BADLEY, JACK...713-497-8599 :94
BANKS JR., GUILBERT ...415-325-9784 :46
BARBER, BRUCE H....713-469-5808 :30
BASTON, BEN P....713-667-6227 :76
BAUMANN, LARRY H....713-498-3433 :87
BECK, TOM R....713-376-7926 :22
BELANGER, FRANK R....713-782-4695 :39
BELLOWS, FRANK...713-622-2089 :88
BLACK, DAVID...713-795-4190 :97
BLAHA, GLEN... :8
BLOXSOM, JOSEPH T....713-781-3669 :54
BLUEFARB, RICHARD A....713-777-6499 :40
BOULINE, GEORGE D.... :24
BREAUX JR., WALTER J.... :10
BRISBIN, JOHN...713-972-1218 :44
CASTROW, DR. FRED F...713-774-7433 :59
CHAPMAN, DOUG...713-977-0909 :49
COATES, STEPHEN W....713-522-0660 :96
COHEN, ROY...713-780-8477 :60
COLLINS, ROBERT V....713-495-3777 :18
CONNER, STEVE...713-777-8912 :98
CORNELSEN, W. HOWARD...713-789-6282 :104
CORNWELL, DENNIS A....713-774-0671 :43
COULTER, MICHAEL D....713-383-2786 :107
DAVIDSON, ROBERT...713-771-4980 :13
DENKER, CHARLES R....713-723-5141 :86
DICKSON, DR. JERRY...713-461-8027 :64
DILLENBURG, RONALD K....713-665-3324 :32
DUNCANSON, ROBERT M.... :38
ESSIG, RAY C....713-497-7165 :93
EUBANK, F.W.... :2
FAY, DOUG...713-667-8093 :74
FELLDIN, JEANNE...713-356-2047 :69
FRACHTMAN, MICHAEL E....713-723-2360 :72
FUKUYA, LESLIE...713-988-0386 :41
GESEL, SANDRA L....713-981-1632 :73
GILBRETH, LEE E....713-342-2685 :100
GUILAK, FARZIN...713-932-1014 :78
HAAR, EDWARD...713-781-9564 :77
HAMILTON, FREDERICK...713-785-9540 :50
HANSEL, DOYLE...713-665-3910 :106
HENDERSON, M.C. ...713-643-2064 :52
HENDRIX, REX E....713-498-7413 :92
HOLLEY JR., DON W....713-644-0544 :35
HURD, DAVID...713-499-2787 :58
JAUBERT, F.L.... :33
JOHNSON, JEFF...713-988-0787 :79
JOHNSON, JOLLY M....713-467-5987 :19
JOHNSTON, MIKE ... :31
KELLER, ROBERT M.... :91
KESLENSKY, JERRY P....713-666-1964 :84
KOPIWODA, ALEX...713-526-8041 :68
LELER, WILLIAM ...713-668-6232 :28
LIPSON, NEIL... :67
LOOMIS, KEN... :11
LUNG, JOHN D....713-621-3577 :103
LYTLE, THOMAS...713-668-6940 :6
MARCHAND, DAVID...713-497-7366 :56
MARTIN, MARSHALL D... :26
MC GEE, JAMES P....713-663-6806 :45
MC KINNEY, M.L. ...713-494-7970 :36
MELTON, LEWIS...713-493-1757 :102
MEYERS, CHRIS B....713-668-3949 :101
MILLER, CARL...713-661-1243 :70
MOTT, CHAPMAN ... :37
MUNSEYJR., MAX... :5
NOVAK, DAVID P....713-497-3291 :16
ODOM, JAMES...713-426-3970 :51
PALMER, DONALD H....713-783-6556 :75
PALMQUIST, C. R....713-688-2105 :80
PARFIMOWICZ, DANIEL J.... :12
PEACOCK, TOM L....713-960-8786 :25
PERKINS, RON...713-342-5247 :81
PERRY JR., WILLIAM L....713-723-1520 :6
PETERSON, ROBERT K....713-666-7644 :95
PEYTON, JAMES L....713-489-8767 :14
PHAYER, JOSEPH C....713-358-1444 :47
PORTER, FLOYD R....405-536-0986 :71
RAUN, LAYTON... :17
ROSEN, SUSAN D....713-771-1614 :62
ROSS, FREDERICK...713-249-3421 :55
ROSSEY, TRENT C....713-933-6218 :89
SANDFIELD, ROBERT E.... :85
SEEGER, EDWARD B....713-723-6919 :29
SHURR, LARRY...713-776-2658 :57
SIMONI, RICHARD T.... :15
SMITH, ROBERT D.... :9
SOUTHERLAND, JOSEPH P.... :23
STERRY, CHRIS W.... :3
STOUT, ROBERT B.... :27
TURNER, JACK C....713-393-1885 :4
TURPEN, TRAVIS...713-476-0640 :63
VAN HOOZER, DEWAYNE...713-682-2126 :7
VAN OVEREN, PETER... :1
VAN WART, CHARLES A.... :34
VANDIVER, LARRY... :90
VELA, MARY LOU...713-782-1594 :82
WALKER, DICK ...713-840-0572 :48
WALLS, B. COOPER...713-933-5813 :99
WARREN, DAVE...713-373-0186 :20
WEINSTOCK, MICHAEL D.... :65
WIBKER, WILLIAM A....713-481-4815 :61
WIEDEMER, JOHN D....713-497-4355 :83
WINTER, KEVIN MARK...713-333-3289 :42
WOLFE, CHARLES B....713-781-6565 :53

IF your telephone number is incorrect or missing, tell Richard so that he can make the changes.

How large is the Houston Area Apple Users Group? 107 and growing. Dues are $12/year. Richard Bluefarb is Treasurer. New members are welcome. Feel free to attend a club meeting, or several, and get to know us. Meeting schedule is printed elsewhere in this issue of Apple Barrel.

The interest inventory will be printed next month.

DATA BASE

by: Mad Bomber

In the last installment of this highly informative series, you were given the 'truth' concerning data base systems. You were told that a data base is just a great big heap of different kinds of data and that a Data Base Management System was a program or set of programs which allow access to the data heap. The intent of this installment of 'DATA BASE' is to introduce several concepts in data structures.

When we talk about 'data structures' we really mean two things. First we're talking about the physical storage of the data, and secondly we're talking about the ways in which the data can be accessed. Some data structures I plan on discussing include lists, decks, and queues. Next month we will cover the more advanced data structures involving trees. We will also finally get into the actual internals of DBMS.

To begin our discussion about data structures let's first define the pointer. The pointer is not an index finger but rather an index which points to a data location. To make things easier consider the following program segment

```
10 MAX =5
20 DIM NODE(MAX)
30 HEAD=0:TAIL=0
```

The variables HEAD and TAIL are pointers (indexes) which are used to indicate the beginning and the ending of the array NODE. MAX just sets the maximum number of entries in the array NODE. Since MAX is set to 5 in statement #10 the array will only contain 5 entries. (AH HA!!! you think the Mad Bomber makes a mistake, no?? I know that APPLE allows the use of zeroth entry in an array. For my purposes this entry does not exist! Put that in your pipe and smoke it!!)

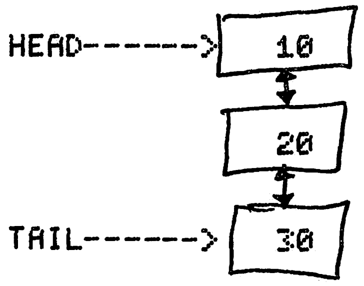
By defination when HEAD is zero then the array is said to be empty. When a pointer is zero it is pointing to nothing.

The list data structure is one of the most commonly used methods of accessing arrays. I will use the list structure to introduce the language notation used in discussing data structures.

In any sturcture the primary concern is to get from the current entry to the next one or to the previous entry without exceeding the bounds of the list as definded by the HEAD and the TAIL. If INX were a pointer to the current entry in the array NODE then INX+1 would point to the next entry and INX-1 would point to the previous entry. When either INX-1 is less than HEAD or INX+1 is greater than TAIL then the bounds of the list have been exceeded.

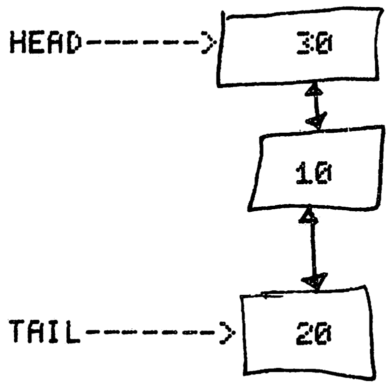
The simple list has two implied pointers. Depending on your point of view of the list these two pointers are called (UP,DOWN) or (LEFT,RIGHT) or (FORWARD,BACKWARD). Let's view NODE as a vertical list structure with the head at the top and the tail at the bottom. If INX points to the current entry then DOWN points to the next entry toward the tail. UP would then point to the next entry toward the head. (ie. UP = INX-1 and DOWN = INX+1) Recall that in order to access the value of an array entry you code something like.. VALUE=NODE(INX). The value of the next entry would be.. VALUE=NODE(DOWN). The value of the previous entry would be.. VALUE=NODE(UP).

One way of viewing data structures is graphically. The diagram of the simple list structure is...

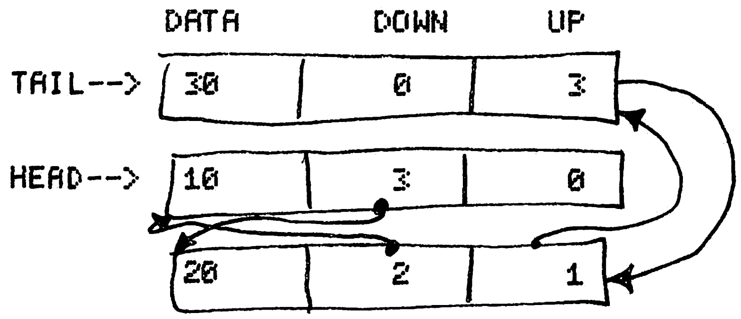


The values of HEAD and TAIL directly index the list. The lines between the boxes represent implied links which connect each entry with the next entry. The between node links are implied because we know that in order to go down from the current entry you add one. To go up you subtract one. The numbers inside the squares are the values of the array at that node.

Notice that if you started at HEAD and printed every node through TAIL you would get the contents of the array in ascending sequence. Suppose you had the following list....



There is no way to print the contents in ascending order. What is needed is a redesign of the links between each node. By making the UP/DOWN links explicit we can provide a path by which the contents of the array can be accessed in order. This is called the doublly linked list structure. The program on the next page uses this structure to keep track of the 'N' largest numbers in an input stream.



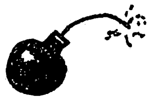
Notice that the down pointer of the tail is zero and the up pointer of the head is also zero. A zero pointer means there is no where to go.

If the up pointer of the head pointed to the tail and the down pointer of the tail pointed to the head then we would have a circle. This circular data structure is called a queue.

A deck data structure is sometimes called a stack. For you people who do assembly language programming you should know all there is to know about stacks. A deck data structure is a LIFO (Last In First Out) list. To visualize a deck think of the discard pile in a card game. You can only retrieve the last card which was put onto the pile. After the top of the discard pile is 'poped' (ie. taken off the pile) then the next card becomes available.

That's too much for this month. Next month by popular request I may tell you how to tell the trees from the forest (if I can find my way out that is).

BOMBS AWAY....



```
]

]LIST

100  REM
110  REM  ---THIS IS A DEMO PGM FOR AN INSERTATION SORT WHICH
120  REM     KEEPS TRACK OF THE TOP-N ITEMS.
350  PRINT "INSERTION SORT EXAMPLE"
360  PRINT "ENTER THE NUMBER OF ITEMS TO KEEP TRACK OF "
370  INPUT Y
380  IF Y < 3 THEN 360
390 M = Y
395 H = 0:T = 0:F = 1
400  PRINT "ENTER THE UNIQUE END OF DATA NUMBER "
410  INPUT E
500  REM
505  DIM X(M),U(M),D(M)
510  REM
520  INPUT Y
530  IF Y = E THEN 5000
540  REM
550  IF T = 0 THEN 1000
560  IF Y > X(T) THEN 1000
565  IF F <  > 0 THEN 1000
580  REM
590  GOTO 520
600  REM
610  REM
1000  GOSUB 2000
1010 X(N) = Y
1020  GOSUB 3000
1030  GOTO 520
1040  REM
1050  REM
2000  IF F <  > 0 THEN 2500
2010 N = T
2020 T = U(T)
2030 D(T) = 0
2040  RETURN
2500 N = F
2510 F = F + 1
2520  IF F > M THEN F = 0
2530  RETURN
2540  REM
2550  REM
3000  IF H <  > 0 THEN 03050
3010 H = N
3020 T = N
3030  RETURN
3040  REM
3045  REM
3050  IF X(N) <  = X(H) THEN 03100
3060 U(H) = N
3070 D(N) = H
3080 U(N) = 0
3090 H = N
3095  RETURN
3097  REM
3098  REM
3100  IF X(N) > X(T) THEN 03200
3110 D(T) = N
3120 U(N) = T
3130 D(N) = 0
3140 T = N
3150  RETURN
3160  REM
```

```
3200 I = H
3210  IF I = 0 THEN  RETURN
3220  REM
3230  REM
3240  IF X(I) < X(N) THEN 03400
3250  IF X(D(I)) > X(N) THEN 3400
3260 D(N) = D(I)
3270 U(D(I)) = N
3280 D(I) = N
3290 U(N) = I
3300  RETURN
3400 I = D(I)
3410  GOTO 3210
4000  REM
4010  REM
4020  REM
5000  PRINT
5010  PRINT
5020  PRINT "WOULD YOU LIKE TO SEE THE DATA IN ASCENDING OR DESCENDING ORDER (A/D)?
5030  INPUT A$
5040  IF A$ = "A" THEN  GOTO 7000
6000 I = H
6010  PRINT X(I)
6020 I = D(I)
6030  IF I = 0 THEN 8000
6040  GOTO 6010
7000 I = T
7010  PRINT X(I)
7020 I = U(I)
7030  IF I = 0 THEN 8000
7040  GOTO 7010
8000  END
```

(this pgm is Available from the HAAUG software Library. Ed)

```
]REM H- POINTER TO HEAD

]REM T- POINTER TO TAIL

]REM F- POINTER TO NEXT FREE NODE

]REM E- END OF DATA INDICATOR

]REM N- POINTER TO NEW NODE

]REM U(I)- UP POINTER FOR NODE(I)

]REM D(I)- DOWN POINTER FOR NODE(I)

]REM X- THE ARRAY WHERE THE LIST IS STORED

]REM M- MAXIMUM SIZE OF ARRAY X

]RUN
INSERTION SORT EXAMPLE
ENTER THE NUMBER OF ITEMS TO KEEP TRACK OF
?3
ENTER THE UNIQUE END OF DATA NUMBER
?-1
?5           ?-1
?
?REENTER
?7           WOULD YOU LIKE TO SEE THE DATA IN ASCENDING OR DESCENDING ORDER (A/D)?
?3           ?D
?8           9
?9           8
```

Ed Seeger, Editor
APPLE BARREL
4331 Nenana Drive
Houston, Texas 77035

(713) 723-6919

```
*************************
*                       *
*  [redacted] CLASS MAIL  *
*  THIRD                *
*************************
```

FRANK BELLOWS
3405 MEADOWLAKE LN.
HOUSTON, TEX. 77027

Postmasters:

<u>Form 3579 requested</u>