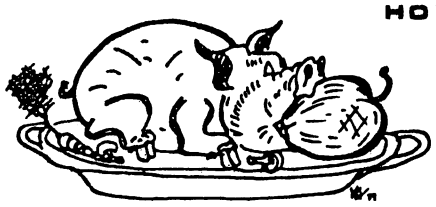
HOUSTON AREA APPLE USERS GROUP



News from the Apple Barrel

==

VOLUME 3 NO. 3 MARCH - APRIL, 1980

President, Bruce Barber Vice President, Bob Stout

Editor, Ed Seeger LEE Gilbreth 342-2685

==

<<< DOUBLE ISSUE >>>

It finally happened!! There is more to publish this month than there is really room for. I think you will enjoy what is here: articles, programs, reviews, announcements, ads. This issue is coming to you under our new bulk mail permit, which means more work to get it ready for the Post Office, and a slower delivery, but at a saving in postage. HAAUG now has 230 members and continues to grow. We pledge ourselves to print information for our numerous beginners and to offer material for the more advanced user as well, such as Lee Meador's DOS 3.2 Disassembly.

Starting this month, APPLE BARREL will reprint Lee Meador's "Disassembly of DOS 3.2," which he has been writing for the FWAUG Newsletter from Fort Worth. Lee, this is the stuff a lot of us have been looking for! I trust you will be pleased to see it getting a wider circulation!

Disassembly of DOS 3.2

Lee Meador

When you put a diskette in your disk drive and boot it up, you might type 6 control-P under the monitor or you might type PR#6. Either way that is the signal for the Apple processor to jump to the machine language program stored in the ROM on the peripheral card in slot 6. That program resides at $C600. The code on the ROM does several things.

First, it constructs a table to translate the bytes on the disk surface into nibbles. (I'll try to explain nibbles later.) Then, it figures out what slot the card is plugged into. Third, it moves the head in the disk drive out to track 0. (See the DOS manual for an explaination of the physical layout.) Next the program finds sector zero and loads the data from that sector into a two part nibble buffer. Part 1 is at $300 and part 2 is at $800. Finally, convert the two buffers of nibbles into one page (256 bytes) of real memory bytes. The last byte is not totally converted.

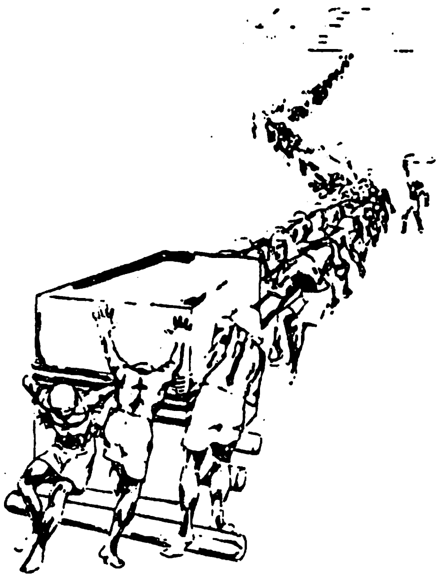
When all this is done the routine in ROM jumps to location $301 if everything has worked correctly.

Now we begin what I call part 2 or stage 2 of the boot or the $300 boot. There are 3 parts to the bootstrap process. We will see part 3 in later month's installments. The $300 boot routine reads the data from track 0, sectors 0-9 into memory at $B600 to $BFFF in a 48K machine. Subtract $4000 or $8000 for 32K and 16K machines respectively. A master diskette always assumes there is 16K of memory for this part of the boot. Thus, a master diskette will load this from $3600 to $3FFF. The $300 boot calls a subroutine in the ROM to load the nibble buffers from the disk surface and then a subroutine at $346 to convert them to real memory bytes. The last thing the $300 boot does is to jump to part 3 of the boot. It is $100 bytes past the beginning of what was loaded from disk. That means it jumps to $B700 on a 48K slave diskette or $3700 on a master diskette.

I promised I would tell you what a nibble was, didn't I? There is a method of recording on magnetic media that uses a Group Recording Code. That means, for example, that 4 bits of data are stored as 5 bits on a tape. This way no two consecutive bits have to be zero. The Apple II Disk uses a similar method that assures several things. No two consecutive bits are zero. Each group of 5 bits is stored as an 8-bit byte on the disk surface. The 7th bit is always a 1 bit. The method used means the data is self synchronizing and thus saves on hardware (and hence pennies for the user.) Now, a nibble is 5 bits that are ready to convert to the 8-bit group code that goes on the disk surface. There is a special way to convert $100 8-bit bytes from memory to $199 5-bit nibbles. You wouldn't believe the method, but you have to. It is outlined on the last few pages.

The stuff that is in track 0, sector 0 is not stored in the usual order. It is shuffled up so the code in ROM will fit into the ROM and still be able to read the disk sector. It is normally kept at $B600 in a 48K apple. You can look at it by booting up and going into the monitor. A 'B600.B6FF' command will show it to you. To see what it says in the right order start at $B6FA. That is the first byte. Then go backward 5 at a time to $B6F5 then $B6F0 and so forth until you get to $B600. Then start over at $B6FB and go back by 5's. When you get to the top of the page go to $B6FC and back. $B6FD and $B6FE begin the next groups. Each group has $33 in it. When you are finished with all 5 columns you need one more byte to fill up your rearranged page. Take the byte at $B6FF and shift it right 3 times (divide by 8). That is the last byte. If you have the patience to do this you will notice that the translation is the same as the stuff at $300 right after a boot. The code at $3D0.3FF is put in on top of the boot routine, but those bytes were all set to $FF in the boot. There is one exception. $3FF is set to $09 in the boot.

The disassembly is commented to show what I think it does. I might be wrong—I was once before. (You might want to know my wife is chuckling over my shoulder.) If you think I'm wrong let me know—directly or indirectly. I do not have access to the original commented assembly language source of these programs.



BY LEE MEADOR - 1401 HILLCREST - ARLINGTON, TX 76010

```
XXXX YYYY FIRST HEX ADDRESS (XXXX) IS THE ADDRESS OF THIS SPOT
          SECOND HEX ADDRESS (YYYY) IS ADDRESS OF A REFERENCE TO THIS
          SPOT. THESE TOGETHER CROSS-REFERENCE THE DISASSEMBLY.

0100 C622 This page is the stack (used to get the slot #)
0300 C6F4 This page is used for part 1 of the nibble buffer
            and, eventually, for the real data (code)
0301 C6F9 Jump to here after the ROM boot is finished
0399 C6DC }<-these are offsets in the 1st nibble buffer used in
03CC C6D8 }   reconstructing the real data
0800 C619 This page is used for part 2 of the nibble buffer
0800 C6AA   and, beginning at $08A0 holds the Disk->Nibble
0800 C6B0   translate table
0800 C6BC
0800 C6CB
0800 C6D4
C080 C63E Access here +0,2,4,6 for the 4 phase off lines }<-used for
C081 C648 Here +0,2,4,6 are the 4 phase on lines         }  seeks
C089 C639 Access here to turn motor on
C08A C636 Access here to enable drive 0 (as opposed to drive 1)
C08C C633 Access here to set Q6 low (read a byte)
C08C C65F
C08C C668
C08C C672
C08C C688
C08C C690
C08C C6A5
C08C C6B7
C08C C6C6
C08E C630 Access here to set Q7 low (set for READ mode)

C600                 CALL HERE TO BEGIN BOOT

C600                                 ;BUILD NIBBLE TRANSLATE TABLE
C600-    A2 20       LDX   #$20      ;Do this for $20 (5 bits) nibbles
C602-    A0 00       LDY   #$00      ;Go back from the end of the page
C604 C60D
C604 C615            LOOP BACK TO HERE IF CURRENT TRY DOESN'T WORK
C604 C61C
C604-    A9 03       LDA   #$03      ;Mask to detect two 0 bits together
C606-    85 3C       STA   $3C       ;
C608-    18          CLC             ;
C609-    88          DEY             ;Try next disk value down
C60A-    98          TYA             ;Now we will test it
C60B C611            LOOP BACK HERE IF CURRENT TRY WORKS SO FAR
C60B-    24 3C       BIT   $3C       ;Are both masked bits = zero
C60D-    F0 F5       BEQ   $C604     ;If so, try next disk value
C60F-    26 3C       ROL   $3C       ;If not, move mask and try again
C611-    90 F8       BCC   $C60B     ; unless we have tried them all
C613-    C0 D5       CPY   #$D5      ;$D5 is a special case (used to
                                     ;  signal beginning of sector)
C615-    F0 ED       BEQ   $C604     ;
C617-    CA          DEX             ;So, we have another "memory" value
C618-    8A          TXA             ;
```

```
C61A-   99 00 08    STA   $0800,Y     ;Put it in the table
C61C-   D0 E6       BNE   $C604       ;If not last "memory" value, loop
C61E                                  ;GET THE SLOT NUMBER OF THIS CARD
C61E-   20 58 FF    JSR   $FF58       ;Does a simple "RTS"
C621-   BA          TSX               ;Look at return address from stack
C622-   BD 00 01    LDA   $0100,X
C625-   48          PHA               ;Save page number (see $C62D)
C626-   0A          ASL               ;Get rid of $C in $Cs of page number
C627-   0A          ASL               , leaving $s0 (s is the slot #)
C628-   0A          ASL
C629-   0A          ASL
C62A-   85 2B       STA   $2B         ;Save that $s0 for future reference
C62C-   AA          TAX               ;And offset Disk Control Addresses
C62D-   A9 D0       LDA   #$D0        ;Sneaky way to put the address $CsD0
C62F-   48          PHA               ; on the stack (see $C625) this is
                                      ; to simulate a return address as
                                      ; if we did a JSR $Cs5D.
C630-   BD 8E C0    LDA   $C08E,X     ;Q7L - set read mode on disk
C633-   BD 8C C0    LDA   $C08C,X     ;Q6L - read a byte, begin reading
C636-   BD 8A C0    LDA   $C08A,X     ;Enable drive 0
C639-   BD 89 C0    LDA   $C089,X     ;Make sure motor is on
                                      ;THIS LOOP MOVES HEAD TO TRACK 0
C63C-   A0 50       LDY   #$50        ;Number larger than the # of tracks
C63E C651           LOOP HERE TO MOVE HEAD DOWN ANOTHER TRACK
C63E-   BD 80 C0    LDA   $C080,X     ;Phase off (the phases move stepper
C641-   98          TYA               ; motor
C642-   29 03       AND   #$03        ;Access one of 4 phase lines (0-3)
C644-   0A          ASL               ; depending on 2 low order bits
C645-   05 2B       ORA   $2B         ; and the slot number (this moves
C647-   AA          TAX               ; the motor a step at a time)3,2,1,0,3,...
C648-   BD 81 C0    LDA   $C081,X     ;Phase on
C64B-   A9 56       LDA   #$56        ;Wait a little bit
C64D-   20 A8 FC    JSR   $FCA8       ; by calling WAIT in monitor
C650-   88          DEY               ;Get ready for next track in
C651-   10 EB       BPL   $C63E       ;Loop if not finished
C653                SET UP ADDRESS TO LOAD PRIMARY NIBBLE PAGE
C653-   A9 03       LDA   #$03        ;$300-$3FF gets part 1 of nibbles
C655-   85 27       STA   $27
C657-   A9 00       LDA   #$00
C659-   85 26       STA   $26
C65B-   85 3D       STA   $3D         ;Load it from sector 0, current track
C65D C67C
C65D C682           FIND GIVEN (IN $3D) SECTOR AND READ IT
C65D C69D             X IS SLOT OFFSET, ($26.27) IS ADDRESS OF PART 1
C65D C6CE             NIBBLE BUFFER, $800 GETS PART 2, $3D IS SECTOR
C65D-   18          CLC               ;Do loop twice, carry clear, then set
C65E C69F
C65E-   08          PHP               ;Loop reacts differently on carry bit
                                      ; CLEAR- finds D5 AA 96 VOL TRK SEC
                                      ; SET  - finds D5 AA AD
C65F C662
C65F C666
C65F-   BD 8C C0    LDA   $C08C,X     ;Read byte from shift register
C662-   10 FB       BPL   $C65F       ;If bit 7 set, shift register
```

```
                                               ; already has byte from disk
C664 C66F
C664-   49 D5       EOR   #$D5       ;Wait for $D5 byte (signals the
C666-   D0 F7       BNE   $C65F      ;  start)
C668 C66B
C668-   BD 8C C0    LDA   $C08C,X    ;Read Another byte
C66B-   10 FB       BPL   $C668      ; (see $C662)
C66D-   C9 AA       CMP   #$AA       ;$AA is second byte
C66F-   D0 F3       BNE   $C664
C671-   EA          NOP              ;Keep timing right
C672 C675
C672-   BD 8C C0    LDA   $C08C,X    ;Read byte
C675-   10 FB       BPL   $C672
C677-   C9 B5       CMP   #$B5       ;$B5 is third byte of sector
C679-   F0 09       BEQ   $C684      ;Skip down if we have a match
C67B-   28          PLP              ;If carry was set at $C65E
                                     ; we don't want a $B5
C67C-   90 DF       BCC   $C65D      ;If clear try again...
C67E-   49 AD       EOR   #$AD       ;If set, we want a $AD, go get data
C680-   F0 1F       BEQ   $C6A1      ;  (see Sector Layout on disk)
C682-   D0 D9       BNE   $C65D      ;If no $AD try again
C684 C679           WE FOUND D5 AA B5, SO READ VOL, TRK SECTOR
C684-   A0 03       LDY   #$03       ;once for vol,twice for trk,
                                     ; three for sec, repeat this loop
C686-   84 2A       STY   $2A        ;Save the 3 for use in loop at C6D2
C688 C68B           READ TWO DISK BYTES FOR ONE 8 BIT VALUE
C688 C698            (SEE FORMAT ROUTINE FOR DETAILS $8F01)
C688-   BD 8C C0    LDA   $C08C,X    ;Read first byte
C68B-   10 FB       BPL   $C688
C68D-   2A          ROL              ;Every other bit here, shifted
C68E-   85 3C       STA   $3C        ;Save it for a moment
C690 C693
C690-   BD 8C C0    LDA   $C08C,X    ;Read second byte
C693-   10 FB       BPL   $C690
C695-   25 3C       AND   $3C        ;Put two parts together
C697-   88          DEY              ;Now read another one of the 3
C698-   D0 EE       BNE   $C688      ;When we leave loop, A has sector
C69A-   28          PLP              ;Get the carry bit off stack
C69B-   C5 3D       CMP   $3D        ;Is this the right sector
C69D-   D0 BE       BNE   $C65D      ;No-try next sector
C69F-   B0 BD       BCS   $C65E      ;Yes-CMP leaves carry set so do
                                     ;  loop with carry set (D5 AA AD)
C6A1 C680           D5 AA B5 VOL TRK SEC D5 AA AD FOUND, NOW READ DATA
C6A1-   A0 9A       LDY   #$9A       ;First read $9A disk bytes
C6A3 C6B3           LOOP TO READ PART 2 NIBBLES ($899-800 IN THAT ORDE
C6A3-   84 3C       STY   $3C
C6A5 C6A8
C6A5-   BC 8C C0    LDY   $C08C,X    ;Read a byte off disk
C6A8-   10 FB       BPL   $C6A5
C6AA-   59 00 08    EOR   $0800,Y    ;Translate to "memory" nibble
C6AD-   A4 3C       LDY   $3C        ;Store in next spot in $0800 page
C6AF-   88          DEY
C6B0-   99 00 08    STA   $0800,Y
C6B3-   D0 EE       BNE   $C6A3      ;If Y<>0 loop for next nibble
```

```
C685 C6C4              LOOP TO READ PART 1 NIBBLE ($00-FF IN PAGE)
C685-    84 3C         STY    $3C
C687 C68A
C687-    BC 8C C0      LDY    $C08C,X    ;Read a byte off disk
C68A-    10 FB         BPL    $C687
C68C-    59 00 08      EOR    $0800,Y    ;Translate to "memory" nibble
C68F-    A4 3C         LDY    $3C        ;Store in next spot in page ...
C6C1-    91 26         STA    ($26),Y    ; defined by ($26.27)
C6C3-    C8            INY               ;Ready for next byte
C6C4-    D0 EF         BNE    $C685      ;Loop if more nibbles needed
C6C6 C6C9
C6C6-    BC 8C C0      LDY    $C08C,X    ;Read just one more byte (as test)
C6C9-    10 FB         BPL    $C6C6
C6CB-    59 00 08      EOR    $0800,Y    ;Compare translation to previous byte
C6CE-    D0 8D         BNE    $C65D      ;If NE we blew it, so try again
C6D0-    60            RTS               ;Nibbles are loaded (see $C62D)
C6D1 C62D and C625 simulate this as return value for $C65D call
C6D1-    A8            TAY               ;Acc is zero
C6D2 C6F2              CONVERT NIBBLES TO ONE PAGE OF REAL DATA ($300...)
C6D2-    A2 00         LDX    #$00       ;page has 5 sections of $33 bytes
C6D4 C6EE
C6D4-    B9 00 08      LDA    $0800,Y    ;part 2 byte has 5 bits right justified
C6D7-    4A            LSR               ;first bit goes with $3CC section
C6D8-    3E CC 03      ROL    $03CC,X
C6DB-    4A            LSR               ;next bit goes with $399 section
C6DC-    3E 99 03      ROL    $0399,X
C6DF-    85 3C         STA    $3C        ;last 3 bits fill corresponding
C6E1-    B1 26         LDA    ($26),Y    ;  byte in $0300 page
C6E3-    0A            ASL               ;  after shifting it into place
C6E4-    0A            ASL
C6E5-    0A            ASL
C6E6-    05 3C         ORA    $3C
C6E8-    91 26         STA    ($26),Y
C6EA-    C8            INY               ;form next byte from nibbles
C6EB-    E8            INX               ;put 3CC & 399 bits in next bytes
C6EC-    E0 33         CPX    #$33       ;Are we to the next section?
C6EE-    D0 E4         BNE    $C6D4      ;No- loop for next conversion
C6F0-    C6 2A         DEC    $2A        ;Was set to 3 in $C686 for 3 sections
C6F2-    D0 DE         BNE    $C6D2      ;If any more sections, loop
C6F4-    CC 00 03      CPY    $0300      ;Y is $99, check if boot is correct
C6F7-    D0 03         BNE    $C6FC      ;If not right boot on disk, skip
C6F9-    4C 01 03      JMP    $0301      ;Begin 2nd boot strap routine
C6FC C6F7
C6FC-    4C 2D FF      JMP    $FF2D      ;Print "ERR" from monitor
C6FF-    FF            .DA    $FF        ;One byte left over on ROM
*    ROUTINES IN THE MONITOR THAT WE CALL
FCA8 C64D Address of WAIT or DELAY routine (call it what you like)
FF2D C6FC Address of PRERR routine in monitor
FF58 C61E Address of IORTS routine (just returns, finds ROM slot)
```

7

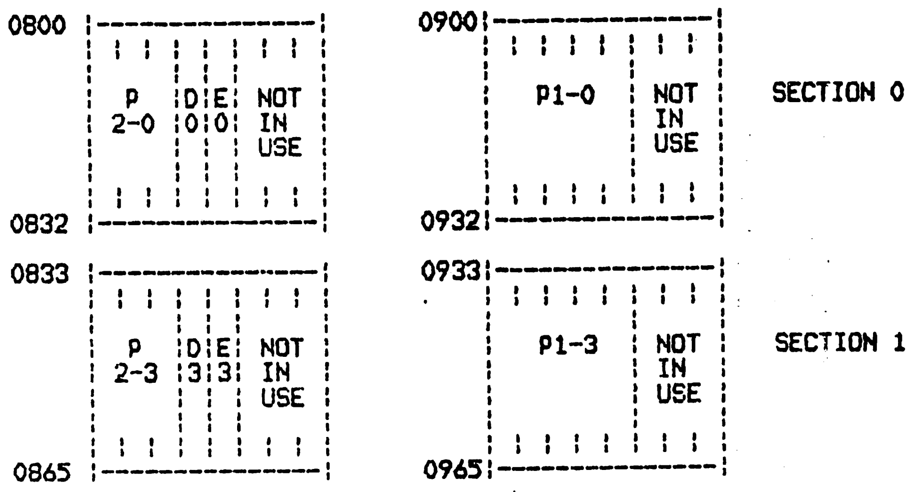
```
003E 0343 Used to jump through in $343
0300-    99             .DA    #$99       ;Used by $C600 boot as validity check

0301 C6F9               BOOT #2--READS IN RWTS, ITS SUBPROGRAMS, AND BOOT
                        THESE RESIDE IN $B600.BFFF (IN 48K APPLE II)

0301 0308               CONVERT NIBBLE TRANSLATE TABLE
0301-    B9 00 08       LDA    $0800,Y    ;Y is $99 at this point
0304-    0A             ASL               ;Load "disk" byte -> "memory"
0305-    0A             ASL               ; nibble translate table entry
0306-    0A             ASL               ; and left justify the nibbles
0307-    99 00 08       STA    $0800,Y    ;The table is in $08A0-08FF
030A-    C8             INY               ;Next table entry
030B-    D0 F4          BNE    $0301      ;Loop if not to end
                        SET UP PARAMETERS FOR CALL TO $Cs5D (assume 48K slave)
030D-    A6 2B          LDX    $2B        ;X gets the slot number ($s0)
030F-    A9 09          LDA    #$09       ;Set page # to load part 1 nibbles
0311-    85 27          STA    $27        ; (note: $26 is already zero)
0313-    AD CC 03       LDA    $03CC      ;This is set by INIT to page #
0316-    85 41          STA    $41        ; of the special stuff ($B600)
0318-    84 40          STY    $40        ;Y is zero right now
031A-    8A             TXA               ;Slot # of form $s0
031B-    4A             LSR               ;Shift to have $0s
031C-    4A             LSR
031D-    4A             LSR
031E-    4A             LSR
031F-    09 C0          ORA    #$C0       ;And form $Cs
0321-    85 3F          STA    $3F        ; which is saved for jumping to
0323-    A9 5D          LDA    #$5D       ; along with $5D (that means we
0325-    85 3E          STA    $3E        ; jump to $Cs5D through $343)
0327 0338               LOOP TO READ PAGES $B6-BF FROM TRACK 0, SECTORS 0-9
0327-    20 43 03       JSR    $0343      ;(JSR $Cs5D) read disk sector
032A-    20 46 03       JSR    $0346      ;Form nibbles into real bytes
032D-    A5 3D          LDA    $3D        ;Get sector number
032F-    4D FF 03       EOR    $03FF      ;Is this the last one
0332-    F0 06          BEQ    $033A      ;Yes-skip down to leave loop
0334-    E6 41          INC    $41        ;Increment page number to load
0336-    E6 3D          INC    $3D        ;Increment sector # to read
0338-    D0 ED          BNE    $0327      ;This should always loop
033A 0332               THEY ARE LOADED, SO JUMP TO PART 3 BOOT ($B700)
033A-    85 3E          STA    $3E        ;Acc is now zero
033C-    AD CC 03       LDA    $03CC      ;Get page # of special stuff ($B6)
033F-    85 3F          STA    $3F
0341-    E6 3F          INC    $3F        ;Increment to get part 3 boot ($B7
                                          ; and jump to part 3 boot
0343 0327
0343-    6C 3E 00       JMP    ($003E)    ;Jump Indirect (1 of 2 uses)
0346 032A               SUBROUTINE TO FORM NIBBLES INTO REAL BYTES
0346-    A2 32          LDX    #$32       ;$33 (0-32) nibbles per section
0348-    A0 00          LDY    #$00       ;We are at start of real memory page
034A 0395               LOOP TO FORM ONE BYTE FROM EACH OF 5 SECTIONS
                        OF LEFT JUSTIFIED NIBBLES. PART 1 IS $900.9FF.
                        PART 2 IS $800.899. (SEE DRAWINGS)
034A-    BD 00 08       LDA    $0800,X    ;Get part2, section 0 nibble
```

```
034D-   4A          LSR               ;Get rid of 3 empty bits
034E-   4A          LSR
034F-   4A          LSR
0350-   85 3C       STA   $3C         ;Save to keep rightmost bit
0352-   4A          LSR               ;Get rid of it now
0353-   85 2A       STA   $2A         ;Save again for new rightmost bit
0355-   4A          LSR               ;And get rid of it too
0356-   1D 00 09    ORA   $0900,X     ;Put the 3 bits left with part1,
0359-   91 40       STA   ($40),Y     ; section 0 nibble and save real byte
035B-   C8          INY               ;Ready for next real byte
035C-   BD 33 08    LDA   $0833,X     ;Get part2, section 1 nibble
035F-   4A          LSR               ;Get rid of 3 bits
0360-   4A          LSR
0361-   4A          LSR
0362-   4A          LSR               ;Now, add one bit into section 4
0363-   26 3C       ROL   $3C         ; part 2 equivalent
0365-   4A          LSR               ;And one bit for section 3 part 2
0366-   26 2A       ROL   $2A         ; equivalent
0368-   1D 33 09    ORA   $0933,X     ;Put the last 3 bits with part1,
036B-   91 40       STA   ($40),Y     ; section 1 nibble and save real byte
036D-   C8          INY               ;Ready for next real byte
036E-   BD 66 08    LDA   $0866,X     ;Get part2, section 2 nibble
0371-   4A          LSR               ;Get rid of 3 bits (again)
0372-   4A          LSR
0373-   4A          LSR
0374-   4A          LSR               ;Save one bit to use with section 4
0375-   26 3C       ROL   $3C
0377-   4A          LSR               ;And one bit to use with section 3
0378-   26 2A       ROL   $2A
037A-   1D 66 09    ORA   $0966,X     ;Put final 3 bits with part 1, section
037D-   91 40       STA   ($40),Y     ; 2 nibble and save real byte
037F-   C8          INY               ;Ready for next real byte
0380-   A5 2A       LDA   $2A         ;We have stored up 3 bits from part 2
0382-   29 07       AND   #$07        ; nibbles-isolate them
0384-   1D 99 09    ORA   $0999,X     ;And put with part 1, section 3 nibble
0387-   91 40       STA   ($40),Y     ;Save real byte
0389-   C8          INY               ;Ready for next real byte
038A-   A5 3C       LDA   $3C         ;We have stored the other 3 bits from
038C-   29 07       AND   #$07        ; part 2 nibbles- isolate them
038E-   1D CC 09    ORA   $09CC,X     ;And put with part 1, section 4 nibble
0391-   91 40       STA   ($40),Y     ;Save real byte
0393-   C8          INY               ;Ready for nex t real byte
0394-   CA          DEX               ;Back up offset into nibble sections
0395-   10 B3       BPL   $034A       ;Keep looping $33 times
0397-   AD 99 08    LDA   $0899       ;Get the "last" nibble, part 2
039A-   4A          LSR               ;Get rid of 3 bits (as always)
039B-   4A          LSR
039C-   4A          LSR
039D-   0D FF 09    ORA   $09FF       ;Add them to part 1 nibbles
03A0-   91 40       STA   ($40),Y     ;Save real "last" byte
03A2-   A6 2B       LDX   $2B         ;Set X to the slot number ($s0)
03A4-   60          RTS               ;We're through
03A5-   FF FF FF    FF'S TO NEXT ADDRESS
03CC 0313
```

```
03CC 033C
03CC-    36           .DA    $B6       ;Page # of special page ($36 on
                                       ; master diskette) Set in $876B.
03CD-    FF FF FF     FF'S TO NEXT ADDRESS
03FF 032F
03FF     09           .DA    $09       ;The "last" byte was not processed
                                       ; in $Cs00 boot. So this is just
                                       ; the part 1 nibble of $48 from
                                       ; 8776 which is $09, the last sect
                                       ; to be read off track 1 in this byte
0800 0301 Used to convert nibble translate table (left justify)
0800 0307
0800 034A Used as part 2 of nibble buffer, section 0
0833 035C  and section 1
0866 036E  and section 2
0899 0397  and "last" nibble
0900 0356 Used as part 1 of nibble buffer, section 0
0933 0368  and section 1
0966 037A  and section 2
0999 0384  and section 3
09CC 038E  and section 4
09FF 039D  and "last" nibble
```

PICTORIAL REPRESENTATION OF NIBBLE —> REAL BYTE CONVERSION

BEGIN:



```
0866 !----------------!    0966!--------------!
     ! ! ! ! ! ! ! !  !        ! ! ! ! ! ! !  !
     !   P  !D!E! NOT !        !  P1-6  ! NOT !    SECTION 2
     !  2-6 !6!6! IN  !        !        ! IN  !
     !      ! ! ! USE !        !        ! USE !
     ! ! ! ! ! ! ! !  !        ! ! ! ! ! ! !  !
0898 !----------------!    0998!--------------!
0899 !----------------!    0999!--------------!
     !niu! P2-L! niu  !        ! ! ! ! ! ! !  !
     !----------------!        !  P1-9  ! NOT !    SECTION 3
     !                !        !        ! IN  !
     !    NO DATA     !        !        ! USE !
     ! ! ! ! ! ! ! !  !        ! ! ! ! ! ! !  !
08CB !----------------!    09CB!--------------!
08CC !----------------!    09CC!--------------!
     ! ! ! ! ! ! ! !  !        ! ! ! ! ! ! !  !
     !                !        !  P1-C  ! NOT !    SECTION 4
     !    NO DATA     !        !        ! IN  !
     !                !        !        ! USE !
     ! ! ! ! ! ! ! !  !        ! ! ! ! ! ! !  !
08FE !----------------!    09FE!--------------!
08FF !----------------!    09FF!--------------!
     ! NO DATA        !        !P1-last ! NIU !    LAST BYTE
     !----------------!        !--------------!

INTERMEDIATE:

---- !----------------!
     ! ! ! ! ! ! ! !  !
     !  P1-0    !  P  !        SECTION 0
     !          ! 2-0 !
     ! ! ! ! ! ! ! !  !
     !----------------!
```

```
+33  !----------------!
     ! ! ! ! ! ! ! !  !
     !  P1-3    !  P  !        SECTION 1
     !          ! 2-3 !
     ! ! ! ! ! ! ! !  !
     !----------------!
+66  !----------------!
     ! ! ! ! ! ! ! !  !
     !  P1-6    !  P  !        SECTION 2
     !          ! 2-6 !
     ! ! ! ! ! ! ! !  !
     !----------------!
+99  !----------------!
     ! ! ! ! ! ! ! !  !
     !  P1-9    !E!E!E!        SECTION 3
     !          !0!3!6!
     ! ! ! ! ! ! ! !  !
     !----------------!
+CC  !----------------!
     ! ! ! ! ! ! ! !  !
     !  P1-C    !D!D!D!        SECTION 4
     !          !0!3!6!
     ! ! ! ! ! ! ! !  !
     !----------------!
LAST !----------------!
     ! P1-LAST ! PS-L !        "LAST" BYTE
     !----------------!
```

FINALLY THE BYTES ARE PUT IN THIS ORDER:

```
+00   SECTION 0, OFFSET $32
+01   SECTION 1, OFFSET $32
+02   SECTION 2, OFFSET $32
+03   SECTION 3, OFFSET $32
+04   SECTION 4, OFFSET $32
+05   SECTION 0, OFFSET $31
+06   SECTION 1, OFFSET $31
+07   SECTION 2, OFFSET $31
 .        .    3,   .    $31
 .        .    4,   .    $31
 .        .    0,   .    $30
 .        .    1,   .    $30
 .        .    .    .     .
 .        .    .    .     .
 .        .    .    .     .
+FA   SECTION 0, OFFSET $00
+FB   SECTION 1, OFFSET $00
+FC   SECTION 2, OFFSET $00
+FD   SECTION 3, OFFSET $00
+FE   SECTION 4, OFFSET $00
+FF   LAST BYTE
```

ALPHABETIZE DISK DIRECTORY

by CHRIS MEYERS

```
]LIST

1000  TEXT : HOME
1010  HIMEM: 32767
1020  POKE 788,1: POKE 789,96: POKE
790,1: POKE 791,0: POKE 792,
0: POKE 793,2: POKE 794,37: POKE
795,3: POKE 796,0: POKE 797,
10: POKE 798,76: POKE 799,1:
 POKE 800,1: POKE 801,136: POKE
802,1: POKE 803,96: POKE 804
,1: POKE 805,0: POKE 806,1: POKE
807,239: POKE 808,216
1030  POKE 768,8: POKE 769,72: POKE
770,152: POKE 771,72: POKE 7
72,138: POKE 773,72: POKE 77
4,160: POKE 775,20: POKE 776
,169: POKE 777,3: POKE 778,3
2: POKE 779,217: POKE 780,3:
 POKE 781,104: POKE 782,170:
 POKE 783,104: POKE 784,168:
 POKE 785,104: POKE 786,40: POKE
787,96
1040 BUFF = 32768: PRINT "LOADING
 DISK DIRECTORY INTO MEM"
1050  FOR I = 12 TO 1 STEP  - 1
1060  POKE 792,17: POKE 793,I
1070  POKE 796, INT ((BUFF / 256 -
 INT (BUFF / 256)) * 256 + .
05): POKE 797, INT (BUFF / 2
56)
1080  CALL 768:BUFF = BUFF + 256
1090  NEXT I:L = 0
1100  HOME : PRINT "FINDING FILES
"
1110  DIM A$(85),DA%(85,5):BUFF =
32768: POKE 34,3: POKE 35,10
: HOME
1120  FOR I = 12 TO 1 STEP  - 1: IF
 PEEK (BUFF + 1) = 0 AND  PEEK
(BUFF + 2) = 0 THEN 1240
1140  FOR K = 11 TO 221 STEP 35: IF
 PEEK (BUFF + K) = 255 THEN
1220
1160 L = L + 1: FOR J = K + 3 + B
UFF TO K + 32 + BUFF: IF  PEEK
(J) = 0 THEN L = L - 1: GOTO
1220
1180 A$(L) = A$(L) +  CHR$ ( PEEK
(J) - 128): NEXT J:DA%(L,1) =
 PEEK (K + BUFF):DA%(L,2) =
 PEEK (K + BUFF + 1):DA%(L,3
) =  PEEK (K + BUFF + 2):DA%
(L,4) =  PEEK (K + BUFF + 33
):DA%(L,5) =  PEEK (K + BUFF
 + 34)
1210GOSUB3000
1220NEXT K
1230 BUFF = BUFF + 256
1240  NEXT I
1250  PRINT : PRINT "THERE ARE ";
L;" FILES ON THIS DISK": FOR
I = 1 TO 1000: NEXT I
1260 PA = 1: TEXT : HOME
1270  VTAB 1: PRINT "SORTING ---
PASS #";PA:SW = 0
1280  FOR I = 1 TO L - 1
1290  IF A$(I) <  = A$(I + 1) THEN
1320
1300  FOR K = 1 TO 5:DA%(85,K) =
DA%(I,K): NEXT K: FOR K = 1 TO
5:DA%(I,K) = DA%(I + 1,K): NEXT
K: FOR K = 1 TO 5:DA%(I + 1,
K) = DA%(85,K): NEXT K
1310 A$(85) = A$(I):A$(I) = A$(I +
1):A$(I + 1) = A$(85):SW = 1

1320  NEXT I
1330  IF SW = 1 THEN PA = PA + 1:
 GOTO 1270
1340  HOME : PRINT "FILES SORTED
--- SAVING TO BUFFER"
1350 BUFF = 32768:L = 0
1360  FOR I = 12 TO 1 STEP  - 1
1370  FOR J = 11 TO 221 STEP 35
1380 L = L + 1: IF A$(L) = "" THEN
1530
1390  VTAB 3: PRINT "WRITING FILE
 #";L;" TO BUFFER"
1400  FOR K = 1 TO  LEN (A$(L))
1410  POKE BUFF + J + K + 2, ASC
( MID$ (A$(L),K,1)) + 128
1420  NEXT K
1430  POKE BUFF + J,DA%(L,1): POKE
BUFF + J + 1,DA%(L,2): POKE
BUFF + J + 2,DA%(L,3): POKE
BUFF + J + 33,DA%(L,4): POKE
BUFF + J + 34,DA%(L,5)
1440  NEXT J
1450 BUFF = BUFF + 256: NEXT I
1460  HOME : PRINT "WRITING DIREC
TORY BACK TO DISK"
1470 BUFF = 32768: FOR I = 12 TO
1 STEP  - 1: POKE 800,2
1480  POKE 792,17: POKE 093,I
1490  POKE 796, INT ((BUFF / 256 -
 INT (BUFF / 256)) * 256 + .
05): POKE 797, INT (BUFF / 2
56)
```

```
1500  CALL 768:BUFF = BUFF + 256
1510  NEXT I:L = 0
1520  PRINT "*** FINISHED SEQUENC
ING DIRECTORY ***": END
1530  VTAB 3: PRINT "CLEARING RES
T OF BUFFER (";84 - L;" FILE
S LEFT) "
1540  FOR K = J TO J + 34: POKE B
UFF + K,0: NEXT K: GOTO 1440

3000 KK = DA%(L,3): IF KK > 127 THEN
 PRINT "*";:KK = KK - 128
3005  IF DA%(L,3) < 128 THEN  PRINT
" ";
3010  IF KK = 0 THEN  PRINT "T";
3020  IF KK = 1 THEN  PRINT "I";
3030  IF KK = 2 THEN  PRINT "A";
3040  IF KK = 4 THEN  PRINT "B";
3050  PRINT " ";:KK = DA%(L,4): IF
KK < 10 THEN  PRINT "0";
3060  IF KK < 100 THEN  PRINT "0"
;
3070  PRINT KK;" ";A$(L)
3080  RETURN
```

Chris makes several comments about his program. He has written his own bubble sort, based on "something I once read, I think it was in Kilobaud or somewhere..." If he had a machine language sort, he would use it, but it cannot be Alan G. Hill's Ampersort, because Chris does not own an Applesoft board, and so cannot use the "&" employed by Hill's sort. If any HAAUG members have or know of a suitable machine language sort that Chris could have, please let him know. Or if you could sell him an Applesoft board very reasonably, that would help. Chris is an eighth grader at Lanier Junior High in Houston.

Might I suggest, Chris, that you change your line 1520 to

```
1520 D$=CHR$(4):  PRINT D$; "CATALOG":  END
```

Then users could see immediately that your program works and their directory is, indeed, in alphabetical order for easier reference.

Many HAAUG members have asked about what it takes to do graphics with the PAPER TIGER. Herb Crosby offers this routine, published here hot off the SOURCE, with listing and directions. If the odd comments and prompts scattered around the page intrigue you, then you'll just have to find out what the SOURCE is! See the article reprinted in the February "Apple Barrel".

```
Off At 21:23 03/08/80
Connect Mins = 1
Compute Secs = 1/0
301 38 DISCONNECTED 0:0:42 30 33

@
)

?SYNTAX ERROR
)PR#00





Welcome to THE SOURCE                Prime System 2.9
QUIT

>mail
Send, Read or Scan:  read

   From:  CL0095            Posted:  Tue  19-Feb-80  0:11  Sys 10  (11)
Subject:  GRAPHICS PROG FOR PAPER TIGER

--More--

1 POKE -16304,1:POKE -16297,0:POKE -16302,0
6 PRINT CHR$(4);"PR#1":PRINT CHR$(9);"255N"
8 PRINT CHR$(17);CHR$(29);CHR$(3) : GOTO 100
20 A=INT(LN/64) : B=INT(LN/8.) : C=INT(8.*(LN/8.-INT(LN/8.))+.001)-A
25 P=(8192+(A*40)+(B*128)+(C*1024)) : RETURN
100 FOR I=0 TO 39
110 J=192
120 LN=J:GOSUB 20
130 O=255-PEEK(P+I):IF O =3 THEN PRINT CHR$(O);
140 PRINT CHR$(O); : J=J-1 : IF J>0 THEN GOTO 120
150 PRINT CHR$(3) : NEXT I : PRINT CHR$(4);"PR#0" : TEXT :END

Disposition:  delete

   From:  CL0095            Posted:  Tue  19-Feb-80  0:16  Sys 10  (8)
Subject:  GRAPHICS FOR PAPER TIGER

--More--

JUST A QUICK AND EASY PROG. PIX SHOULD BE LOADED INTO PAGE ONE (HGR)
THEN RUN THIS PROG.  PIX WILL DISPLAY ON TV WHILE PRINTER IS PRINTING
IT AND WILL GO BACK TO NORMAL AFTER PIX IS DUMP TO PAPER TIGER....
ONE DOT ON SCREEN = ONE DOT ON PAPER. PRODUCES 3.5 BY 5 INCH PIX
TO DO PAGE TWO 8192 MUST BE CHANGED AND PROPER POKE TO DISPLAY IT
INVERSE PIX BY CHANGING 130 TO O=PEEK INSTEAD OF 255-PEEK
THIS PROG RUNS IN APPLE SOFT AND IS SLOW
HERBERT

Disposition:  delete

   From:  TCD434            Posted:  Sat  1-Mar-80  14:21  Sys 10  (11)
Subject:  EIN BRIEF

--More--no
Disposition:  deletge

   From:  TCA455            Posted:  Thu  6-Mar-80  2:39  Sys 10  (48)
```

HOUSTON NEEDS A MAIL ORDER COMPUTER DISCOUNT STORE

Al Sevcik

Why do we, living in Houston, buy disk drives and printers through Fred Fuchs' fabulous pipeline to New Jersey? Why do we buy RAM chips through HAAUG's California Special?

Cost, of course. Mail order is often cheaper.

It's true that each of us likes a good price deal, but we also appreciate having a place to go when our hardware hardly performs. The local computer merchants, however, maintain that they can't match these discount deals and still support showroom, technical salespeople, and a handy fix-it shop.

Question: Is it in the best interests of HAAUG membership to support purchases by mail order from out of town?

Answer: (YOU fill in the blank.)

MY answer is an unequivocal, "Yes." And further, I think the local stores are missing a good bet. I believe there exists a minority of personal computer owners who will endure the time, trouble and tears of ordering by mail to save bucks. This is not lost business to Houston stores because their list price shops were never up for consideration. This money settles strictly on the low price deal, wherever it may be.

Any local store owner who understands the concept of marginal sales volume, and who sets up a low profile department that offers mail order discounts and discounts for quantity purchases by groups would, I believe, be well rewarded. Discreet advertising in club newsletters and/a strict "by mail only" ordering policy would control the overhead. Buyers would get what they want without the long delays of out-of-town mail delivery.

The majority of customers, businessmen who want service and immediate performance, and individuals who need hand-holding to effect a purchase would not be attracted. But the money now being mailed to New Jersey and California would be diverted to Houston shops.

Everybody wins.

Okay Mr. or Ms Computer Store Owner. The next step is yours. I'll be looking for your ad.

HAAUG member Al Sevcik's challenge speaks for itself. If there is an Apple dealer among the readership, who would like to reflect on the realities of retailing, his or her comments will be published in an early issue. If each member of HAAUG has invested an average of, say, $2500 in an Apple system, then our membership represents some half-million plus dollars already spent! Is it true that this money has sought the lowest price, or conversely, is it true that lowered prices are the dealers' major bait for attracting business? Should W. Bell & Company offer discounted Apples, along with the Samsonite luggage and the Minoltas? Again, a responsible reply is solicited.

* * * HAAUG's ABBS 654-0759 * * *
Herb's ABBS 480-1840

Here are two ads taken off the club's Apple Bulletin Board System. If you do not have a modem, think seriously about saving for one. After your first disk drive, a modem just may be the next most useful peripheral to get. It's not for nothing that D.C. Hayes has taken out a 4-page color center-fold ad in the APPLE ORCHARD. MicroNET, the SOURCE, our own ABBS, and numerous other bulletin board and timesharing systems are available to you through your Apple. Several HAAUGs are now transfering software and hi-res pictures over the telephone, using Ed Magnin's outstanding programs. Ed does business as the Telephone Software Connection. If you want to see what a modem can mean, call (213) 329-6548 from Applesoft some evening and download one of his free demos, while surveying the menu of goodies. You will be amazed what your Apple can do!

MSG # 20
SUBJ.: AD
TO: ED SEEGER
FROM: COOPER WALLS
D TE: 02/26/80

MR. SEEGER,
I'D LIKE TO PUT A WANT AD IN THE NEXT APPLE BARREL. THE TEXT IS IN MESSAGE #14 FROM ME. THANK YOU.
8:20

MSG# TO RETRIEVE (1/23), 14

MSG # 14
SUBJ.: APPLESOFT CARD
TO: ALL
FROM: COOPER WALLS
DATE: 02/22/80

I'VE UPGRADED TO A PASCAL SYSTEM AND NO LONGER CAN USE MY OLD APPLESOFT CARD. IF YOU WANT TO BUY IT CALL ME, COOPER WALLS, 713-933-5813 EVENINGS.

MSG # 20
SUBJ.: ROBOTICS
TO: ANYONE
FROM: FRED FUCHS
DATE: 03/17/80

IS ANYONE ELSE INTERESTED IN BUILDING A ROBOT?
I HAVE INVESTIGATED MANY OF THE BOOKS BUT
I SIMPLY DO NOT HAVE ALL THE KNOWLEDGE OR FUNDS TO
COMPLETE THE PROJECT. I AM GOING TO TRY TO BUILD
A ROBOT WITH BOTH SELF CONTROL AND REMOTE CONTROL.
I INTEND TO USE ONE OF TWO POSSIBLE TYPES OF DATA LINKS
ONE POSSIBILITY IS TO USE THE ULTRASONIC TRANCEIVER I
ALREADY HAVE. THE OTHER, WOULD BE TO USE A RADIO SYSTEM
OF SOME TYPE. ALTHOUGH THIS COULD PRESENT SOME PROBLEMS
AS I DO NOT HAVE A HAM TICKET. I HAVE THOUGHT ABOUT CB
BUT THE INTERFERENCE WOULD PROBABLY PREVENT THIS. I ALSO
WISH TO PROVIDE AT LEAST SOME FORM OF DUPLEXING SO THAT
THE CONTROLER CAN HAVE SOME IDEA WHAT THE ROBOT IS ACTUALLY
DOING. IN ANY CASE, I HOPE TO LOCATE SOMEONE ELSE WITH
THE DESIRE TO BUILD

* * * PASCAL INTRODUCTION AND PROJECT * * *

-- by Pat McGee

HAAUG will hold a course in USCD Pascal for members only. The course will be designed for people with some knowledge of programming. If demand warrants, remedial sessions for beginners will be available. All aspects of the Apple Pascal system will be covered. Use of the assembler will be covered in separate sessions.

The course will start in June, at a time and place to be arranged at our mutual convenience. Each session will be about two hours long. We will meet once a week for at least eight weeks. The instructor is Pat McGee. There will be assigned readings, homework, and a class project. You must have access to an Apple Pascal system.

The class project will be placed in the HAAUG software library and should be something useful to a wide audience. We will decide on the project at an early meeting. Some possibilities are a text formatter, a data manager like Whatsit? or File Cabinet, or a personal accounting system.

Class members incur no financial obligations except HAAUG dues. There is no charge for the course. However, you are expected to do and turn in on time every homework assignment and to contribute to the class project. The only good way to learn to program is to program, and have someone give you feedback on how well you are doing.

For further information come to the June HAAUG meeting, or sign up with Pat McGee before then. Please indicate meeting times that are especially convenient, possible, or impossible, as well as areas of town that are good or bad for you.

HOMEWORK #1 DUE AT FIRST CLASS: Think up at least two programs suitable for a class project. Think of something you want your Apple to do, or something you do now that could be done better. Include for each idea what the program should do, why you believe it is a good idea, and how the program would make you (or someone else) happier. Write down how the user would use the program, what he should input to it, what the program should put back to the user, and what it should keep on file until later. List any necessary extra equipment that would be needed.

Put each idea onto a separate piece of paper, typed double spaced, or legibly written. Bring them to the first class with you.

==> (You may wonder what relation, if any, this course bears to David Black's, which is outlined elsewhere in this issue. Pat's is open only to HAAUG members and carries no fee. It's goals are more limited; it will not cover PASCAL to the extent David Black / Computer City's will. Much of Pat's course will focus on creating a class project, whereas David's aims to impart a thorough understanding of the language and will be using class time to that end. Pat himself cautions that "you get what you pay for!" If you have a true commitment to learning and using PASCAL, taking BOTH courses might be an excellent route to follow, for one will teach the language in depth, while the other will afford practice in applying it to solving a problem/project.)

HAAUG PASCAL USERS DIRECTORY

Joe E. Saiz	943-0192
Robert Sandfield	871-0023
Fred H. Fuchs	781-6968
David P. Novak	522-1781 (office/day)
Mike McKinney	933-2447
James Odom	426-3970 (modem)
Alex Kopiwoda	821-2702
Jim Castrow	465-1748
Ed Seeger	723-6919
Pat McGee	666-0004
Lynn Evans	790-4493 (office)
Robert Collins	495-3777

Each of the above has differing levels of skill in programming PASCAL, of course, so this directory is not to be taken as anything more than a listing of who else shares your interest. If you need help, go ahead and make a few calls. If you GET help, write up what you needed and what you learned and APPLE BARREL will publish it.

Call Ed Seeger if you'd like to be added to the directory.

BAVID BLACK / COMPUTER CITY

INTRODUCTION TO PROGRAMMING IN PASCAL

COURSE OUTLINE

I. AN INTRODUCTION TO THE UCSD 'PASCAL' LANGUAGE SYSTEM
*1. OVERVIEW
A. INTRODUCTION
1. WHO AM I & HOW CAN I BE REACHED?
2. HOMEWORK & READING
3. STUDENTS' BACKGROUND
B. WHAT IS 'PASCAL'?
1. HIGH LEVEL LANGUAGE (HLL) DEVELOPED BY NIKALAUS WIRTH (VIRTH), AND DISTRIBUTED BY GRADUATE STUDENTS
2. ADVANTAGES OVER 'BASIC'
A) FASTER EXECUTION
B) LARGER PROGRAMS & SMALLER GO-CODE
C) NO PENALTY FOR COMMENTS & FREE FORMAT
D) STRUCTURED CONSTRUCTS
E) ADVANCED DATA STRUCTURES (EX. RECORDS)
F) ADVANCED SUBROUTINE/FUNCTION MECHANISM (=> LANG. EXTENSION)
G) STRICT DATA TYPING (AVOID ERRORS)
3. DISADVANTAGES
A) COMPILATION BEFORE EXECUTION (=> CATCH SYNTAX)
B) FORCED DECLARATION (=> THINK)
C) LITTLE DYNAMIC STORAGE ALLOCATION
D) LESS DIRECT MACHINE CONTROL (=> INDEPENDENCE & PORTABILITY)
C. ENHANCEMENTS
1. UCSD ADDITIONS TO 'STANDARD PASCAL'
A) CHARACTER STRING DATA TYPE
B) RELAXATION OF SOME RESTRICTIONS
C) TURTLE-GRAPHICS
D) FILE MANAGEMENT
E) EXTENDED PRECISION NOS. (BCD)
F) MACRO-ASSEMBLER INTERFACE
D. EXAMPLES OF 'PASCAL'
1. WRITE A MESSAGE
2. DRAW & LABEL A BOX
3. COUNTING TO TEN
4. PRIME NUMBERS
5. GRAFDEMO
E. A BREAKDOWN OF THE SYSTEM
1. GETTING STARTED
2. THE COMMAND-LINE
3. CTL-Z, CTL-A, <-, ->
4. EXECUTION OF PROGRAMS
5. FILER
A) WHAT IS A FILE?
B) DEVICES
6. EDITOR
A) WHAT IS A TEXT-FILE?
B) FORMAT
C) SYSTEM.WORK.TEXT
7. COMPILER
A) WHAT IS A COMPILER VS. INTERPRETER?

B) ADVANTAGES
C) DISADVANTAGES
8. ASSEMBLER
A) MACHINE LANGUAGE INTERFACE
B) SUPER FAST CODE!
9. DEBUGGER
A) RUN-TIME DIAGNOSTICS
B) IMPLEMENTATION
II. ELEMENTARY PROGRAMMING IN 'PASCAL'
*2. THE FIRST STEPS
A. EDITING
1. INVOCATION
2. COMMAND-LINE
3. INSERT
4. DELETE
5. CURSOR MOVEMENT
6. EXITING THE EDITOR
B. FILING
1. SETTING TODAY'S DATE
2. SAVING THE WORKFILE
3. LISTING THE DIRECTORY
4. LISTING ONLINE DEVICES
5. PRINTING A TEXT FILE
6. EXITING THE FILER
C. COMPILING
1. R)UN
2. FIXUP
D. EXECUTION
1. X)ECUTE VS. R)UN
2. RUN-TIME ERRORS
E. EXAMPLES
1. WRITE A MESSAGE
2. COUNT TO TEN
*3. PROGRAM FLOW
A. PROGRAM FORMAT
1. 'PROGRAM'
2. 'LABEL'
3. 'CONST' - COMPILE TIME SYMBOLS
4. 'TYPE' - INTEGER,REAL,ARRAY,RECORD
5. 'VAR' - REQUIRED
6. 'PROCEDURE' & 'FUNCTION'
7. BODY: 'BEGIN' .. 'END'
8. SEMICOLONS & FINAL 'END.'
B. SIMPLE I/O
1. WRITELN('TEXT')
2. WRITE & WRITELN
3. READ(V)
4. READLN & READ
5. EXAMPLES
C. SIMPLE VARIABLES :INTEGER
1. RECALL INTEGER BASIC OR % VARIABLES
2. MAXINT
3. DECLARATION
4. EXAMPLE:

```
PROGRAM XYZ;
  CONST R=10;
  VAR X,Y,Z:INTEGER;
      I:INTEGER;
  BEGIN READ(X,Y,Z);
        WRITE('THANKS...');
        I := X + Y + Z;
        WRITELN(10*I)
  END.
```

```
        5. OPERATORS
           A) ARITH
           B) LOGIC
     D. LOOPING
        1. FOR <VAR> := <EXPR> TO <EXPR> DO <STMT>
        2. WHILE <COND> DO <STMT>
        3. REPEAT <STMT> UNTIL <COND>
        4. EXAMPLES
     E. SELECTION
        1. IF <COND> THEN <STMT>
        2. IF <COND> THEN <STMT> ELSE <STMT>
        3. CASE <EXPR> OF <CASELIST> END
        4. EXAMPLES
*4. SUBROUTINES & SCOPE OF IDENTIFIERS
     A. MORE VARIABLE TYPES :REAL,BOOLEAN,CHAR,STRING
        1. REAL
        2. BOOLEAN: TRUE, FALSE
        3. SINGLE CHARACTER
        4. STRING
        5. LONG INTEGERS
        6. EXAMPLES
     B. PROCEDURES
        1. ANALOGY WITH 'GOSUB'
        2. LANGUAGE EXTENSIONS
        3. STRUCTURE
        4. SIMPLE EXAMPLES
        5. INVOCATION
        6. PARAMETERS
        7. USE IN PROGRAM DEVELOPMENT
     C. PARAMETERS
        0. SCOPE OF VARIABLES
        1. BY VALUE
        2. BY REFERENCE (MENTION ONLY)
        3. LIMITING COMMUNICATION
        4. EXAMPLES
     D. FUNCTIONS
        1. ANALOGY WITH 'DEF FN'
        2. DEFINITION
        3. RETURNING . VALUE
        4. INVOCATION
        5. EXAMPLES
     E. EXAMPLES
```

III. COMPLETE 'PASCAL'
*5. MORE SUBROUTINES
A. REVIEW OF SUBROUTINES & PARAMETERS
1. A NATURAL WAY TO REDUCE PROGRAM COMPLEXITY
2. DISADVANTAGES OF THE LESSON
3. OVERALL PROCEDURE STRUCTURE
4. MORE ON PARAMETERS
A) BY VALUE
B) BY REFERENCE
C) DATA TYPES
D) EXAMPLES
B. NESTING OF PROCEDURES
1. WHO CALLS WHO
2. RESTRICTING FUNCTIONALITY TO IMPROVE DEBUGGING
3. HIDING DETAILS
C. SCOPE OF VARIABLES W.R.T. NESTING
1. LOCAL VARIABLES
2. 'EXTERNAL' VARIABLES
3. GLOBAL VARIABLES
4. RESTRICTING COMMUNICATION
D. RECURSION (=> SIMPLE CONCEPTUAL IMPLEMENTATION)
1. EXAMPLE
2. WHAT HAPPENS
3. MORE EXAMPLES
4. MISUSE
E. EXAMPLES
*6. PROGRAM STRUCTURE & DEVELOPMENT
A. UNDERSTANDING THE PROBLEM
1. GET A WRITTEN DESCRIPTION
2. LET SOMEONE ELSE READ IT
3. IS IT SPECIFIC OR VAGUE?
B. SPECIFY INPUT & OUTPUT
1. WHAT ARE THE INPUTS & OUTPUTS
2. WHAT FORMATS
3. ARE YOU CONSIDERING HUMAN ENGINEERING
C. DIVIDE & CONQUER STRATEGY
1. OUTLINE THE ATTACK STRATEGY OR USE CHARTS.
2. USE PROCEDURE CALLS
3. PSUEDO-CODE TO LEAVE OUT DETAILS
D. TOP DOWN DEVELOPMENT
1. LEAVE DETAILS UNTIL LAST
2. POSTPONE DATA STRUCTURE DECISIONS
E. EXAMPLES
*7. DATA STRUCTURES I.
A. REVIEW
B. DATA TYPES
C. VARIABLES
D. CONSTANTS
E. ARRAYS
F. EXAMPLES
*8. DATA STRUCTURES II.
A. SETS
B. EXAMPLES
C. RECORDS

```
    D. EXAMPLES
    E. SORTING
*9. INPUT/OUTPUT
    A. SIMPLE I/O
    B. FORMATTED
    C. SEQUENTIAL FILES
    D. EXAMPLES
    E. DIRECT ACCESS (RANDOM)
    F. EXAMPLES
*10.MISCELLANEOUS & APPLE SPECIFICS        (TO BE COVERED ONLY IF THERE IS
    A. GOTO STMT                            SUFFICIENT TIME AVAILABLE)
    B. TURTLE GRAPHICS
    C. APPLESTUFF
    D. SCIENTIFIC MATH
    E. BUSINESS APPLICATIONS
    F. ADDING TO YOUR LIBRARY
*11.DATA STRUCTURES III.                          ""          ""
    A. POINTERS
    B. LINEAR LISTS
    C. RINGS
    D. TREES
    E. EXAMPLES
```

HAAUG members have by now received a mailing that announces Dave Black's PASCAL programming course. The course outline, prepared, by the way, on the Apple USCD PASCAL's text editor, is printed above. This is clearly a sophisticated, in-depth course for persons with a serious interest in learning the language. Dave's course is being offered through Computer City, 821-2702.

BOOK REVIEW

"A PRIMER ON PASCAL", RICHARD CONWAY, DAVID GRIES, E. CARL ZIMMERMAN, WINTHROP PUBLISHERS, INC., CAMBRIDGE, MA., 433 PP., $10.95

AS THE TITLE IMPLIES, THIS BOOK IS A PRIMER ON USE OF THE PASCAL LANGUAGE. THE FIRST FEW CHAPTERS ARE VERY SIMPLE AS THEY ARE DESIGNED FOR A USER WITH NO PRIOR PROGRAMMING EXPERIENCE. HOWEVER, FOR ONE WHO IS NOT A COMPUTER PROFESSIONAL AND WHOSE ONLY PROGRAMMING EXPERIENCE IS USING THE APPLE BASICS THIS BOOK PROVIDES AN EXCELLENT INTRODUCTION TO PASCAL.

A FEATURE WHICH I CONSIDER EXCELLENT IS THAT IN THE VERY FIRST CHAPTER PROGRAMMING EXAMPLES FOR TRIVIAL PROBLEMS ARE GIVEN WHICH DO NOT GET BOGGED DOWN IN LONG DISCUSSIONS ON THE PASCAL SYNTAX WHICH TEND TO TURN OFF THE NOVICE USER. IN THIS WAY ONE IS INTERACTING WITH THE APPLE FROM THE START. AFTER EACH PROGRAM LISTING THERE IS A DISCUSSION SECTION WHICH POINTS OUT THE IMPORTANT FEATURES OF THE PROGRAM, INCLUDING SYNTAX DIAGRAMS AS NEEDED.

THIS BOOK HAS FIVE PARTS STARTING WITH FUNDAMENTAL CONCEPTS WHICH INCLUDES DISCUSSIONS AND PROGRAM EXAMPLES FOR PASCAL VARIABLES, ASSIGNMENT STATEMENTS, COMPOUND STATEMENTS, LABEL DECLARATIONS, ETC.. IT THEN PROGRESSES THROUGH SECTIONS WHICH INCLUDE PROGRAM STRUCTURE AND DEVELOPMENT, SUBPROGRAMS (PROCEDURES), TESTING AND RUNNING.

THERE ARE PROBLEMS AT THE END OF EACH CHAPTER, BUT NO ANSWERS GIVEN IN THE BOOK. THE INDEX IS EXCELLENT AND OF GREAT HELP WHEN TRYING TO DEBUG OR UNDERSTAND A PROGRAM.

A PRIMER ON PASCAL IS NOT A REPLACEMENT FOR THE APPLE PASCAL MANUAL AS IT DOES NOT DEAL WITH SPECIAL FEATURES OF APPLE'S IMPLEMENTATION OF THE UCSD PASCAL. HOWEVER, THE PROGRAM ILLUSTRATIONS COMPILE AND RUN ON THE APPLE WITH NO PROBLEM.

I STRONGLY RECOMMEND THIS BOOK FOR SOMEONE WHO HAS RECENTLY PURCHASED APPLE'S LANGUAGE SYSTEM AND IS FRUSTRATED WITH TRYING TO WRITE PASCAL PROGRAMS.
-RUDGE ALLEN

* * * ASK DR. APPLE * * *

Dear Dr. Apple,

I'm having difficulty tabbing past column 40 on my printer. i have tried to poke location 36 as done in File Cabinet, but I destroy my program after printing the first one (which does print correctly). I am using a Heath H14 serial printer with the SSM AIO interface card. What can I try?

Signed, Frustrated

Dear Frustrated,

When using the HTAB command to position a line on the printer, you can tab to the limit of your printer. However, afdter the line goes to the printer it also goes to the screen, which in most cases has a shorter line width. A tab beyond the limit of the screen will destroy an area of memory which holds pointers. The result will be unpredictable problems such as hang-ups, wrong positioning, blowing your program, etc. You mention that you are using the SSM AIO interface to drive your printer. The AIO allows you to disable the video screen. The proper POKE to do this is slot-dependent, so consult your owner's manual. This POKE will bypass the video routine which caused your problem.

-- Dr. Apple

Dr. Apple has a backlog of other queries, which will be addressed in upcoming issues. HAAUG members are asking him about PEEKS, POKES, AND CALLS, conversion of programs from TRS-80 BASIC to Apple's BASIC, machine language programming, HIRES text screen, etc. etc. Keep asking. And remember, "An Apple a day gives the doctor his say!"

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

(713) 723-6919

```
**************************
*       BULK RATE        *
*   THIRD CLASS MAIL     *
*                        *
**************************
```

C

H.A.A.U.G

Postmasters:

Address correction requested

DeWayne Van Hoozer
4510 Avalon
Lawton, OK 73501