AMPERDHR

DOS 3.3
O
C

If you have a //c or //e with an extended 80-column card, the world of double Hi-Res graphics awaits you. These ampersand graphics utilities allow you to use this new graphics format directly from Applesoft by simulating the normal Hi-Res graphics commands.

by Steven Meuse

The advent of double Hi-Res graphics offers exciting possibilities for the graphics programmer. Whether you're looking for video excitement or business graphics, you can count on sharper, more realistic graphics once you master the challenge of double Hi-Res programming. This article and a demonstration program will enable you to experiment easily with double Hi-Res using the standard Hi-Res commands you already know.

To join our adventure in double Hi-Res, you should have an Apple //c or //e with an extended 80-column text card for a total of 128K RAM, and a jumper connected across the two Molex-type connectors near the keyboard end of the card.

To install the jumper on the extended 80-column card, first get the actual jumper that came with the card (it should be taped to the instruction sheet on how to install the jumper). If, like me, you didn't get a jumper, you'll need an alligator clip or something like it. The pins themselves are about a quarter of an inch back from the keyboard end of the extended 80-column card, near the bottom as you look at it inside the Apple. They will either point straight away from the card (toward slot 7) or be turned 90 degrees and point toward the keyboard. Connect the jumper or alligator clip across the two pins, and you're all set. You can leave the jumper on from now on if you like; it shouldn't affect the normal operation of your Apple.

WHEN IS 8K NOT 8K?

Figure 1 shows the memory map for the //c and //e with an extended 80-column text card. The 64K RAM on the text card in the //e (called auxiliary memory) essentially mirrors the main 64K RAM that comes in the //e (let's call this the main memory). In the //c, both main and auxiliary memory are built in. What we are concerned with here is Hi-Res page 1 and its auxiliary counterpart, Hi-Res page 1X. In double Hi-Res mode, the Apple //e's video circuitry scans both of these Hi-Res pages simultaneously and produces a display twice as dense (with twice as much screen memory altogether).

Let's take a closer look at this, using just the top line of a normal Hi-Res screen, from upper left to upper right. It takes 40 bytes to hold the information to produce this line. Of the eight bits in each byte, only seven are actually displayed on the Hi-Res screen as dots, and the eighth bit (the high bit) selects the color of the dots. Seven displayed bits per byte multiplied by 40 bytes gives us 280 bits (or dots) on each line.

In order to get twice the horizontal resolution, twice as much memory is needed to store the picture information. Hi-Res page 1X is used for this, and it effectively gives us a total of 80 bytes across the screen for each line. Again, only seven bits in each byte are displayed, for 7 x 80, or 560 dots on each line.

Thus, the two 8K Hi-Res pages together can store one double Hi-Res picture. The problem is that they both occupy the same address range in memory! Don't worry — using the built-in soft switches, we can select, or "bank in" one page or the other into the actual 64K RAM that the Apple's 6502 microprocessor can address. The trick is to get some meaningful data into Hi-Res page 1X. Unfortunately, Applesoft has never even heard of Hi-Res page 1X. That's where these machine language double Hi-Res routines come in.

TABLE 1: Available DHR Colors

0 - Black	8 - Dark Blue
1 - Magenta	9 - Violet
2 - Brown	10 - Grey2
3 - Orange	11 - Pink
4 - Dark Green	12 - Medium Blue
5 - Grey1	13 - Light Blue
6 - Light Green	14 - Aqua
7 - Yellow	15 - White

THE GOOD STUFF

AMPERDHR offers the Applesoft programmer a way of directly using double Hi-Res graphics through ampersand commands which simulate the normal Applesoft Hi-Res commands.

There are two parts to the AMPERDHR program: the set-up program located on page 3 (**Listing 1**), and the main routines (**Listing 2**), which load into the RAM card area of memory. The set-up program loads the main program in just above Hi-Res page 1, and moves it up in memory to bank 1 of the language card. It also sets the ampersand vector and leaves a few routines behind to manage the memory switching between Applesoft and the main routines. The main routines start at $D000 and include a command parser and the actual code that manipulates the Hi-Res screens.

In order to use the program, first turn the 80-column card on, then load in and call the set-up program as follows:

```
10 REM INITIALIZE AMPERDHR ROUTINES
20 PRINT CHR$(4)"PR#3"
30 PRINT CHR$(4)"BLOAD AMPERDHR.SET
   UP"
40 CALL 790
```

The rest is taken care of by the set-up program. Once it has executed, only the area from $376 to $3CB is still used. You can use $300 to $375 (decimal 768 to 885) for your own programs.

Those of you who are familiar with language cards may have noticed that the program, in using bank 1, shouldn't interfere with anything else in the card. Right! I've tested the routines with Integer BASIC in the card, as well as relocated ProntoDOS (an excellent product, by the way) with no problems.

All of Applesoft's Hi-Res functions are simulated, with the exception of DRAW and

XDRAW. The main reason for their omission is speed. Initial versions of the program were on the slow side, partially because it has twice as much RAM to tend to, and partially due to the amount of memory bank switching needed to access 16K of RAM mapped into 8K of space.

The actual function of the program is largely transparent. The command syntax is identical to Applesoft's, as shown in the demonstration program (**Listing 3**). It uses the ampersand to transfer program control to the command parser. The commands may be within a program or entered from the keyboard. They are:

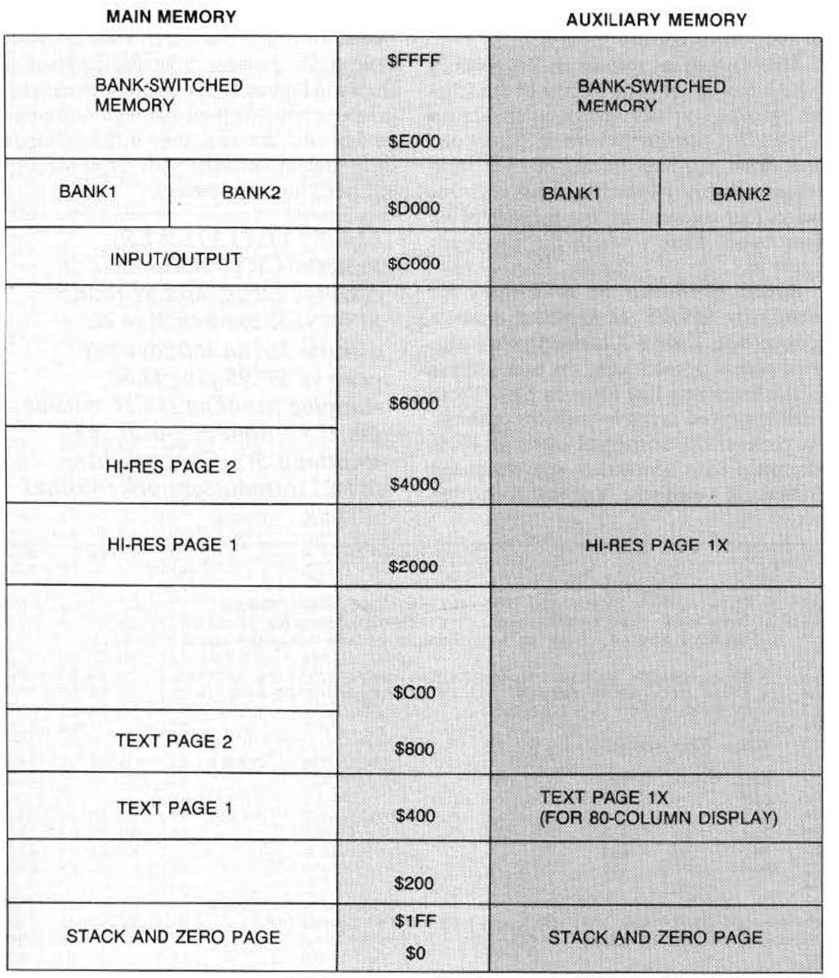
1. &HGR (clears the double Hi-Res screen to black, sets mixed text/graphics mode, sets the color to black)
2. &HCOLOR= (sets the color to 0-15)
3. &HPLOT (the same as Applesoft, except the legal X-range is 0,559; TO works exactly the same)
4. &CLEAR (clears the double Hi-Res screen to the most recent HCOLOR)
5. &COLOR= (sets double medium resolution mode with 1; sets double Hi-Res mode with a 0)
6. &LOAD and &SAVE (used for BLOADing and BSAVEing double Hi-Res pictures)
7. The Applesoft commands TEXT, POKE −16302,0 (for full-screen graphics) and POKE −16301,0 (for mixed text/graphics) still work normally.

THE FINE PRINT

As with standard Hi-Res, there are a few caveats with double Hi-Res. First, use &HGR before any &HPLOTting. Also, set your resolution mode and color before your first plot. Usually, this will mean &COLOR=0 (for Hi-Res mode) and &HCOLOR= (whichever you prefer).

The sixteen available colors along with their corresponding color numbers are shown in **Table 1**. Although these are the same sixteen colors that standard Lo-Res uses, the numbers are not the same.

FIGURE 1: 128K Memory Map



BSAVEing and BLOADing double Hi-Res pictures requires a few more steps than before. DOS has never heard of Hi-Res page 1X either, so the &LOAD and &SAVE commands act as intermediaries and transfer information between the two Hi-Res pages. The syntax for loading and saving double Hi-Res pictures is shown in **lines 760-920** of **Listing 3**.

THE PLOT THICKENS

A short time ago, the words "double medium resolution" popped up, and went right by. I saved this for last because I have to delve into screen mapping and other things in order to explain it. If you do not need to understand why it works, here's the simple explanation: Double medium resolution always draws solid lines, regardless of color. Double Hi-Res plots pixel by pixel, just like standard Hi-Res but with twice the horizontal resolution. Double medium-res lines are thicker, even blocky, but always solid. Think of double medium-res as a graphics mode between double Lo-Res and standard Hi-Res. The vertical resolution is the same as standard Hi-Res (192 lines), but the horizontal resolution is half (140 lines).

Of course, all sixteen colors (actually fifteen, since grey is counted twice) are available. If you've ever been frustrated by drawing a near-vertical colored line and getting many pieces of a line, double medium-res may be for you.

To key in AMPERDHR, first enter the set-up program shown in **Listing 1**. If you have an assembler, the source code may be entered and assembled. Alternatively, you may use the Monitor to enter just the hex code as described in "A Welcome to New *Nibble* Readers" in the beginning of this issue. Save the program on disk with the command.

```
BSAVE AMPERDHR.SETUP,A$316,
L$B6
```

Since the code shown in **Listing 2** is meant to be located in the RAM card area of memory, it must be entered from the Monitor at another location. To key it in, type in the code using a starting address of $4000 instead of $D000 as shown. When the entire program has been entered, save it on disk with the command:

```
BSAVE AMPERDHR.D000,A$4000,
L$558
```

To key in the Applesoft demonstration program, enter the code shown in **Listing 3** and save it on disk with the command:

```
SAVE AMPERDHR.DEMO
```

A BIT OF NITTY-HGRITTY

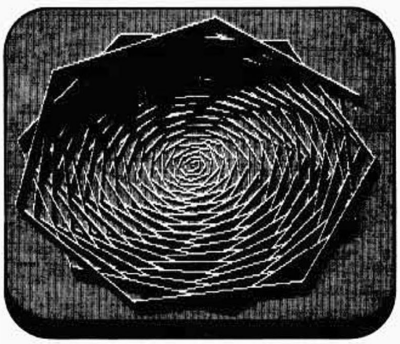
For all of you who do need to know how this works, here goes: Remember how in standard Hi-Res the odd coordinates could only be one of two colors, and the even coordinates worked the same way with two

FIGURE 2: Double Hi-Res Screen Bytes

MEMORY BANK	$2000-AUX								$2000-MAIN								$2001-AUX								$2001-MAIN							
HORIZONTAL COORDINATE	X	6	5	4	3	2	1	0	X	13	12	11	10	9	8	7	X	20	19	18	17	16	15	14	X	27	26	25	24	23	22	21
COLOR GROUP#		2	2	2	1	1	1	1		4	4	3	3	3	3	2		6	5	5	5	5	4	4		7	7	7	7	6	6	6
SAMPLE BYTES	0	0	0	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0

X = UNUSED COLOR BIT

FIGURE 3
Perspectives from AMPERDHR.DEMO



different colors, depending on the color bit? Well, drop the idea of a color bit (although it's still present within each byte, it is now ignored), and put all four of those colors into groups of four contiguous, *displayed* bits — that's about how double Hi-Res works. In other words, the color you see at any point on the screen is now determined by four bits (or pixels) instead of two.

Look at **Figure 2**. As in standard Hi-Res, the Apple's video circuitry reads each screen byte from right to left (from the least significant to the most significant bit), and the high bit is not displayed. The twist is that the double Hi-Res display starts at coordinate 0,0 with a byte from auxiliary memory, then goes to the same address in main memory, then to the next highest address in auxiliary memory, then to that same address in main memory, and so on. So not only are the color-determining groups of four bits not in the same byte, they are often not even in the same memory bank!

The medium-res plotting routines always plot in these groups of four bits. They are interleaved with the Hi-Res routines, and use several look-up tables to keep track of what they are doing. In general, however, the medium-res mode uses a different bit mask (four bits wide instead of one), and it usually requires two separate calls to the actual plotting routine, one for each byte that the group of four bits occupies.

This is not to say that the Hi-Res routines have it any easier. The pattern of the four-bit groups repeats every four screen "bytes." (I use quotes here to signify one byte from our theoretical, 80-byte wide screen memory.) Therefore, each color requires four color masks, one for each of the four bytes. Here's where the speed-ups come in.

Instead of shifting the color mask for every new HPLOT (as Applesoft does), a look-up table is used. A look-up table is also used instead of calculating the base address of the horizontal line for each plot. These have improved speed by a factor of three. At present, the horizontal offset in bytes (from the base address) is still calculated from the X-coordinate. You will notice that horizontal lines are a bit slower to plot than vertical ones.

This is why there are no DRAW or XDRAW commands. They would be uselessly slow, using a technique similar to Applesoft's. I've considered using pre-shifted byte shape tables, and . . . well, what can I say? I had to stop somewhere!

A few demonstrations are included in AMPERDHR.DEMO to get you started. For the most part, these are public domain programs adapted for double Hi-Res. In particular, Double Cross takes about fourteen minutes to complete, but it borders on a transcendental experience in double Hi-Res mode, even on a black-and-white monitor! One of the pictures from Perspectives is shown in **Figure 3**. Try all of the demo programs in both medium and high resolution modes, and see how they differ. Change them around, or write your own. Most of all, have fun with them!

LISTING 1: AMPERDHR.SETUP

```
1     ********************************
2     *      AMPERDHR.SETUP          *
3     *   DOUBLE HI-RES ROUTINES     *
4     *   (SETUP & PG 3 VECTORS)     *
5     *                              *
6     *         (C) 1983             *
7     *      BY STEVE MEUSE          *
8     *                              *
9     *                              *
10    *                              *
11    *                              *
12    *      MERLIN ASSEMBLER        *
13    *                              *
14    ********************************
15
16
17    * This routine loads the DHR routines (AMPERDHR.D000)
18    * into memory at $4000, and calls the monitor move routine
19    * to move the routines up into Bank 1 of the language card.
20
21    * This routine should be executed before installing another
22    * short program on page 3.  After execution, $300-$375 is
23    * available for use (768-885).
24
25    *Zero Page Equates:
26
27    A1L     =    $3C
28    A1H     =    $3D
29    A2L     =    $3E
30    A2H     =    $3F
31    A4L     =    $42
32    A4H     =    $43
33
34
35    * Applesoft routines that will be called by vector
36    * from the language card:
37
```

```
                        38
                        39  GETADR   =    $E752
                        40  DATA     =    $D995
                        41  FRMNUM   =    $DD67
                        42  SYNCHR   =    $DEC0
                        43  GETBYT   =    $E6F8
                        44  IQERR    =    $F199       ;ILLEGAL QUANTITY ERROR
                        45  SNERR    =    $DEC9       ; SYNTAX ERROR
                        46  ADDON    =    $D998
                        47  COUT     =    $FDED
                        48  MOVE     =    $FE2C
                        49
                        50
                        51  *Other Equates:
                        52  PARSER   =    $D000
                        53  LCOFF    =    $C082
                        54  READBK1  =    $C088
                        55  WRITEBK1 =    $C089
                        56  AMPER    =    $3F5
                        57
                        58
                        59            ORG  $316
0316: A9 8D             60            LDA  #$8D         ;Carriage Return
0318: 20 ED FD          61            JSR  COUT
                        62
031B: A2 00             63  MOVEUP   LDX  #$00
031D: BD 5A 03          64  MOVEUP1  LDA  LOADSTR,X
0320: 20 ED FD          65            JSR  COUT
0323: BD 5A 03          66            LDA  LOADSTR,X
0326: E8                67            INX
0327: C9 8D             68            CMP  #$8D
0329: D0 F2             69            BNE  MOVEUP1      ;BLOAD routines thru DOS hooks
                        70
032B: A9 40             71            LDA  #$40
032D: 85 3D             72            STA  A1H
032F: A9 4F             73            LDA  #$4F
0331: 85 3F             74            STA  A2H
0333: A9 D0             75            LDA  #$D0
0335: 85 43             76            STA  A4H
0337: A9 00             77            LDA  #$00
0339: 85 3C             78            STA  A1L
033B: 85 3E             79            STA  A2L
033D: 85 42             80            STA  A4L
033F: A8                81            TAY
0340: 2C 82 C0          82            BIT  LCOFF        ;Insurance- turn card off first
0343: 2C 89 C0          83            BIT  WRITEBK1
0346: 2C 89 C0          84            BIT  WRITEBK1
0349: 20 2C FE          85            JSR  MOVE
034C: 2C 82 C0          86            BIT  LCOFF
                        87
                        88
                        89
034F: A9 76             90  INITAMPR LDA  #<GOCARD
0351: 8D F6 03          91            STA  AMPER+1
0354: A9 03             92            LDA  #>GOCARD
0356: 8D F7 03          93            STA  AMPER+2
0359: 60                94            RTS
035A: 84                95  LOADSTR  HEX  84           ;Control-D
035B: C2 CC CF          96            ASC  "BLOAD AMPERDHR.D000,A$4000"
035E: C1 C4 A0 C1 CD D0 C5 D2
0366: C4 C8 D2 AE C4 B0 B0 B0
036E: AC C1 A4 B4 B0 B0 B0
0375: 8D                97            HEX  8D           ;Carriage Return
                        98
                        99
                        100
                        101
0376: 2C 88 C0          102 GOCARD   BIT  READBK1
0379: 20 00 D0          103           JSR  PARSER
037C: 2C 82 C0          104           BIT  LCOFF
037F: 4C 95 D9          105           JMP  DATA
                        106
0382: 2C 82 C0          107 GETBYTV  BIT  LCOFF
0385: 20 F8 E6          108           JSR  GETBYT
0388: 2C 88 C0          109           BIT  READBK1
038B: 60                110           RTS
                        111
038C: 2C 82 C0          112 IQERRV   BIT  LCOFF
038F: 4C 99 E1          113           JMP  IQERR
                        114
0392: 2C 82 C0          115 ADDONV   BIT  LCOFF
0395: A0 01             116           LDY  #$01
0397: 20 98 D9          117           JSR  ADDON
039A: 2C 88 C0          118           BIT  READBK1
039D: 60                119           RTS
                        120
039E: 2C 82 C0          121 SNERRV   BIT  LCOFF
03A1: 4C C9 DE          122           JMP  SNERR
                        123
03A4: 2C 82 C0          124 COUTV    BIT  LCOFF
03A7: 20 ED FD          125           JSR  COUT
03AA: 2C 88 C0          126           BIT  READBK1
03AD: 60                127           RTS
                        128
03AE: 2C 82 C0          129 SYNCHRV  BIT  LCOFF
03B1: 20 C0 DE          130           JSR  SYNCHR
03B4: 2C 88 C0          131           BIT  READBK1
03B7: 60                132           RTS
```

LISTING 1: AMPERDHR.SETUP *(continued)*

```
                        133
03B8: 2C 82 C0          134 FRMNUMV  BIT  LCOFF
03BB: 20 67 DD          135           JSR  FRMNUM
03BE: 2C 88 C0          136           BIT  READBK1
03C1: 60                137           RTS
                        138
03C2: 2C 82 C0          139 GETADRV  BIT  LCOFF
03C5: 20 52 E7          140           JSR  GETADR
03C8: 2C 88 C0          141           BIT  READBK1
03CB: 60                142           RTS

--End assembly--

182 bytes

Errors: 0
END OF LISTING 1
```

LISTING 2: AMPERDHR.D000

```
                        1   ********************************
                        2   *        AMPERDHR.D000         *
                        3   *    DOUBLE HI-RES ROUTINES     *
                        4   *         MAIN PROGRAM          *
                        5   *                               *
                        6   *        (C) 1983               *
                        7   *      BY STEVE MEUSE           *
                        8   *                               *
                        9   *     MERLIN ASSEMBLER          *
                        10  *                               *
                        11  ********************************
                        12
                        13  * This program ORG's in the unused 4K bank of the language
                        14  * card, and can co-exist w/ relocated DOS or Integer BASIC.
                        15
                        16  * The ampersand hook banks in the card, and gives control
                        17  * to the command parser.  On return, the motherboard ROMs
                        18  * are re-activated, and control is given to DOS and BASIC.
                        19
                        20  MSKPTR   =    $19
                        21  SHAPEL   =    $1A
                        22  SHAPEH   =    $1B
                        23  HCOLOR1  =    $1C
                        24  COUNTH   =    $1D
                        25  GBASL    =    $26
                        26  GBASH    =    $27
                        27  HMASK    =    $30
                        28  A1L      =    $3C         ;Used with AUXMOVE:
                        29  A1H      =    $3D         ;A1 contains source start address
                        30  A2L      =    $3E
                        31  A2H      =    $3F         ;A2 contains source end address
                        32  A4L      =    $42
                        33  A4H      =    $43         ;A4 contains target start address
                        34  LINNUM   =    $50
                        35  DSCTMP   =    $9D
                        36  CHRGET   =    $B1
                        37  CHRGOT   =    $B7
                        38  DXL      =    $D0
                        39  DXH      =    $D1
                        40  DY       =    $D2
                        41  QDRNT    =    $D3
                        42  EL       =    $D4
                        43  EH       =    $D5
                        44  XØL      =    $E0
                        45  XØH      =    $E1
                        46  YØ       =    $E2
                        47  HCOLORZ  =    $E4
                        48  HNDX     =    $E5
                        49  HPAG     =    $E6
                        50  RESMODE  =    $E7         ;used for SCALE7 in normal hires
                        51  PLOT2    =    $F9         ;used for ROTZ in normal hires
                        52  AUXMOVE  =    $C311       ;Carry set = transfer main to aux
                        53
                        54  *    Soft switches:
                        55
                        56  STOR80OF =    $C000
                        57  RAMRD    =    $C002       ;OFF
                        58  RAMWRT   =    $C004       ;OFF
                        59  STORE80  =    $C001
                        60  ALTZP    =    $C008
                        61  AN3      =    $C05E
                        62  COL80    =    $C00D
                        63  PAGE2ON  =    $C055
                        64  PAGE2OFF =    $C054
                        65  TEXT     =    $C050
                        66  MIXSET   =    $C053
                        67  HIRES    =    $C057
                        68
                        69  * Vectors to Applesoft routines:
                        70
                        71  GETBYTV  =    $382
                        72  IQERRV   =    $38C        ;Illegal quantity error
                        73  SNERRV   =    $39E        ;Syntax error
                        74  ADDONV   =    $392        ;Increments TXTPTR
                        75  COUTV    =    $3A4
                        76  SYNCHRV  =    $3AE        ;Syntax checker
                        77  FRMNUMV  =    $3B8        ;returns formulas in FAC
                        78  GETADRV  =    $3C2        ;turns FAC -> 2-byte num in LINNUM
                        79
                        80  * Keyword tokens:
                        81
                        82  hgr      =    145
                        83  hcolor   =    146
                        84  hplot    =    147
                        85  color    =    160
                        86  clear    =    189
                        87  to       =    193
                        88  load     =    182
                        89  save     =    183
                        90
                        91
                        92            ORG  $D000       ;BANK 1
                        93
D000: C9 91             94  PARSER   CMP  #hgr         ;This is the parsing routine.
D002: D0 04             95            BNE  PARSE1
D004: 20 64 D0          96            JSR  DHGR
D007: 60                97            RTS
D008: C9 92             98  PARSE1   CMP  #hcolor
D00A: D0 04             99            BNE  PARSE2
D00C: 20 A7 D1          100           JSR  DHCOLOR
D00F: 60                101           RTS
D010: C9 93             102 PARSE2   CMP  #hplot
D012: D0 04             103           BNE  PARSE3
D014: 20 3D D0          104           JSR  DHPLOT
D017: 60                105           RTS
D018: C9 BD             106 PARSE3   CMP  #clear
D01A: D0 04             107           BNE  PARSE4
D01C: 20 80 D0          108           JSR  DBCKGND
D01F: 60                109           RTS
D020: C9 A0             110 PARSE4   CMP  #color
D022: D0 04             111           BNE  PARSE5
D024: 20 BA D1          112           JSR  DCOLOR
D027: 60                113           RTS
D028: C9 B6             114 PARSE5   CMP  #load
D02A: D0 04             115           BNE  PARSE6
D02C: 20 56 D3          116           JSR  DLOAD
D02F: 60                117           RTS
D030: C9 B7             118 PARSE6   CMP  #save
D032: D0 04             119           BNE  NOPARSE
D034: 20 64 D3          120           JSR  DSAVE
D037: 60                121           RTS
D038: 68                122 NOPARSE  PLA              ;There was no match...
D039: 68                123           PLA              ;Clean up stack
D03A: 4C 9E 03          124           JMP  SNERRV      ;Syntax error
                        125
```

```
D03D: 20 B1 00  126 DHPLOT   JSR  CHRGET
D040: C9 C1     127          CMP  #'to       ;Continued plot requested?
D042: F0 0D     128          BEQ  HP3        ;Branch if so
D044: 20 CF D1  129          JSR  DHFNS      ;Get coor of start point
D047: 20 52 D1  130          JSR  DHPLOT0    ;Plot it, setting up coor
D04A: 20 B7 00  131 HP2      JSR  CHRGOT
D04D: C9 C1     132          CMP  #'to       ;Line specified?
D04F: D0 12     133          BNE  RTS3       ;Exit if not
D051: 20 AE 03  134 HP3      JSR  SYNCHRV
D054: 20 CF D1  135          JSR  DHFNS      ;Get coor of line end
D057: 84 9D     136          STY  DSCTMP     ;Set up for line
D059: A8        137          TAY
D05A: 8A        138          TXA
D05B: A6 9D     139          LDX  DSCTMP
D05D: 20 DA D2  140          JSR  DHLIN      ;Plot line
D060: 4C 4A D0  141          JMP  HP2        ;Loop till no more "TO"
D063: 60        142 RTS3     RTS
                143
D064: A9 20     144 DHGR     LDA  #$20       ;  Hires initialization -  &HGR
D066: 8D 01 C0  145          STA  STORE80    ;Assumes a "PR#3" has been issued
D069: 8D 0D C0  146          STA  COL80      ;Note that the usual commands
D06C: 8D 57 C0  147          STA  HIRES      ;POKE-16302,0 (full screen)
D06F: 8D 5F C0  148          STA  AN3        ;POKE-16301,0 (mixed screen)
D072: 8D 53 C0  149          STA  MIXSET     ;and TEXT still work as before
D075: 85 E6     150          STA  HPAG
D077: 8D 50 C0  151          STA  TEXT
D07A: A9 00     152          LDA  #$00
D07C: 85 E4     153          STA  HCOLORZ
D07E: 85 1C     154          STA  HCOLOR1
                155
D080: A9 20     156 DBCKGND  LDA  #$20       ;  Hires clear routine -  &CLEAR
D082: 85 1B     157          STA  SHAPEH
D084: A0 00     158          LDY  #$00
D086: 84 1A     159          STY  SHAPEL
D088: A5 E4     160          LDA  HCOLORZ
D08A: AA        161          TAX
D08B: BD B8 D0  162 DBCKGND1 LDA  BACKCOLR,X
D08E: 8D 55 C0  163          STA  PAGE2ON
D091: 91 1A     164          STA  (SHAPEL),Y
D093: 8D 54 C0  165          STA  PAGE2OFF
D096: BD B9 D0  166          LDA  BACKCOLR+1,X
D099: 91 1A     167          STA  (SHAPEL),Y
D09B: C8        168          INY
D09C: BD BA D0  169 DBCKGND2 LDA  BACKCOLR+2,X
D09F: 8D 55 C0  170          STA  PAGE2ON
```

LISTING 2: AMPERDHR.D000 *(continued)*

```
D0A2: 91 1A     171          STA  (SHAPEL),Y
D0A4: BD BB D0  172          LDA  BACKCOLR+3,X
D0A7: 8D 54 C0  173          STA  PAGE2OFF
D0AA: 91 1A     174          STA  (SHAPEL),Y
D0AC: C8        175          INY
D0AD: D0 DC     176          BNE  DBCKGND1
D0AF: E6 1B     177          INC  SHAPEH
D0B1: A5 1B     178          LDA  SHAPEH
D0B3: 29 1F     179          AND  #$1F
D0B5: D0 D4     180          BNE  DBCKGND1
D0B7: 60        181          RTS
D0B8: 00 00 00  182 BACKCOLR HEX  000000000
D0BB: 00
D0BC: 08 11 22  183          DFB  %00001000,%00010001,%00100010,%01000100
D0BF: 44
D0C0: 44 08 11  184          DFB  %01000100,%00001000,%00010001,%00100010
D0C3: 22
D0C4: 4C 19 33  185          DFB  %01001100,%00011001,%00110011,%01100110
D0C7: 66
D0C8: 22 44 08  186          DFB  %00100010,%01000100,%00001000,%00010001
D0CB: 11
D0CC: 2A 55 2A  187          DFB  %00101010,%01010101,%00101010,%01010101
D0CF: 55
D0D0: 66 4C 19  188          DFB  %01100110,%01001100,%00011001,%00110011
D0D3: 33
D0D4: 6E 5D 3B  189          DFB  %01101110,%01011101,%00111011,%01110111
D0D7: 77
D0D8: 11 22 44  190          DFB  %00010001,%00100010,%01000100,%00001000
D0DB: 08
D0DC: 19 33 66  191          DFB  %00011001,%00110011,%01100110,%01001100
D0DF: 4C
D0E0: 55 2A 55  192          DFB  %01010101,%00101010,%01010101,%00101010
D0E3: 2A
D0E4: 5D 3B 77  193          DFB  %01011101,%00111011,%01110111,%01101110
D0E7: 6E
D0E8: 33 66 4C  194          DFB  %00110011,%01100110,%01001100,%00011001
D0EB: 19
D0EC: 3B 77 6E  195          DFB  %00111011,%01110111,%01101110,%01011101
D0EF: 5D
D0F0: 77 6E 5D  196          DFB  %01110111,%01101110,%01011101,%00111011
D0F3: 3B
D0F4: 7F 7F 7F  197          DFB  %01111111,%01111111,%01111111,%01111111
D0F7: 7F
                198
                199 * HIRES PLOT SUBROUTINES:
                200
D0F8: 85 E2     201 DHPOSN   STA  Y0         ;Enter with Y-coor in Accumulator
D0FA: 86 E0     202          STX  X0L        ;Lo byte of X-coor in X-register
D0FC: 84 E1     203          STY  X0H        ;Hi byte of X-coor in Y-register
D0FE: A8        204          TAY
D0FF: B9 47 D4  205          LDA  YLOOKUPL,Y ;Get base address of Y-coor from
D102: 85 26     206          STA  GBASL      ;lookup table ard store in GBAS
D104: B9 87 D3  207          LDA  YLOOKUPH,Y
D107: 85 27     208          STA  GBASH
D109: 8A        209          TXA             ;using X-coor
D10A: A4 E1     210          LDY  X0H        ;Calc. offset from base address
D10C: 38        211          SEC
                212                          ;Divide X0 by 7
D10D: F0 18     213          BEQ  DHPOSN4    ;to get screen
D10F: C0 02     214          CPY  #$02       ;column number (0-79)
D111: F0 0D     215          BEQ  DHPOSN1
D113: 18        216          CLC
D114: A0 23     217          LDY  #$23
D116: 69 04     218          ADC  #$04
D118: 90 0B     219          BCC  DHPOSN2
D11A: C8        220          INY
D11B: E9 07     221          SBC  #$07
D11D: 90 06     222          BCC  DHPOSN2    ;Always
D11F: 00        223          BRK             ;Just in case!
```

```
D120: A0 48     224 DHPOSN1  LDY  #$48
D122: 18        225          CLC
D123: 69 01     226          ADC  #$01
D125: 38        227 DHPOSN2  SEC
D126: C8        228 DHPOSN3  INY
D127: E9 07     229 DHPOSN4  SBC  #$07
D129: B0 FB     230          BCS  DHPOSN3
D12B: 84 E5     231          STY  HNDX       ;Got "byte" offset from base 0-79
D12D: 69 07     232          ADC  #$07       ;Make the remainder positive
D12F: 85 30     233          STA  HMASK      ;Keep it around
D131: 98        234 DHPOSN5  TYA             ;Get HNDX back
D132: 29 03     235          AND  #$03       ;Divide MOD 4
D134: A8        236          TAY             ;for possible MEDRES1 use later
D135: 18        237          CLC
D136: 65 E4     238          ADC  HCOLORZ    ;add HCOLOR * 4 to it
D138: 85 1C     239          STA  HCOLOR1    ;store color index pointer
D13A: 24 E7     240          BIT  RESMODE    ;which mode are we in?
D13C: 30 01     241          BMI  MEDRES1    ;Medium res- keep going
D13E: 60        242          RTS             ;Hires- that's it for now
D13F: B9 54 D5  243 MEDRES1  LDA  BITTABL,Y
D142: 65 30     244          ADC  HMASK      ;Calculate pixel offset from the
D144: AA        245          TAX             ;last column evenly divisible by 4
D145: 85 19     246          STA  MSKPTR
D147: BD 1C D5  247          LDA  MSKTBPTR,X ; (offset will equal 0-27)
D14A: 85 30     248          STA  HMASK      ;get the mask pointer (7-20)
D14C: BD 38 D5  249          LDA  PLOT2FLG,X ;get second plot flag from table
D14F: 85 F9     250          STA  PLOT2      ;store it
D151: 60        251          RTS
                252
D152: 20 F8 D0  253 DHPLOT0  JSR  DHPOSN     ;got coors, get addresses ,etc
D155: A5 E5     254 DHPLOT1  LDA  HNDX       ;Get horiz. index (0-79)
D157: 4A        255          LSR             ;Divide by 2
D158: B0 03     256          BCS  DHPLOT2    ;screen "byte" is odd- main mem
D15A: 8D 55 C0  257          STA  PAGE2ON    ;even- use aux mem
D15D: A8        258 DHPLOT2  TAY             ;now use proper mapping- (0-39)
D15E: A6 1C     259          LDX  HCOLOR1    ;get pointer to color mask table
D160: BD B8 D0  260          LDA  BACKCOLR,X ;get color mask for this byte
D163: 51 26     261          EOR  (GBASL),Y  ;EOR with hires screen
D165: A6 30     262          LDX  HMASK      ;get coordinate mask pointer
D167: 3D 07 D5  263          AND  MSKTABL,X  ;AND accumulator w/coordinate mask
D16A: 51 26     264          EOR  (GBASL),Y  ;EOR it again
D16C: 91 26     265          STA  (GBASL),Y  ;store it in screen memory
D16E: 8D 54 C0  266          STA  PAGE2OFF   ;always leave main mem on!
D171: 24 E7     267          BIT  RESMODE    ;are we in Medium res?
D173: 10 06     268          BPL  RTS1       ;Nope, we're done
D175: 24 F9     269          BIT  PLOT2      ;Yes, any more plotting to do?
D177: 30 03     270          BMI  UPBYTE     ;Yes, the next highest "byte"
D179: 70 15     271          BVS  DOWNBYTE   ;Yes, the next lowest "byte"
D17B: 60        272 RTS1     RTS             ;No more plotting left
D17C: 46 E7     273 UPBYTE   LSR  RESMODE    ;Fool DHPLOT1 to return here
D17E: E6 E5     274          INC  HNDX       ;increment all pointers
D180: E6 30     275          INC  HMASK
D182: E6 1C     276          INC  HCOLOR1
D184: 20 55 D1  277          JSR  DHPLOT1    ;plot the second point
D187: C6 E5     278          DEC  HNDX       ;return pointers to normal
D189: C6 30     279          DEC  HMASK
D18B: C6 1C     280          DEC  HCOLOR1
D18D: 06 E7     281          ASL  RESMODE    ;RESMODE too
D18F: 60        282          RTS             ;finished
D190: 46 E7     283 DOWNBYTE LSR  RESMODE    ;same as UPBYTE
D192: C6 30     284          DEC  HMASK
D194: C6 1C     285          DEC  HCOLOR1
D196: C6 E5     286          DEC  HNDX
D198: 20 55 D1  287          JSR  DHPLOT1
D19B: E6 E5     288          INC  HNDX
D19D: E6 30     289          INC  HMASK
D19F: E6 1C     290          INC  HCOLOR1
D1A1: 06 E7     291          ASL  RESMODE
D1A3: 60        292          RTS
                293
D1A4: 4C 8C 03  294 GGERR    JMP  IQERRV     ;illegal quantity error
                295
D1A7: 20 92 03  296 DHCOLOR  JSR  ADDONV     ; Set hires color -  &HCOLOR
D1AA: 20 82 03  297          JSR  GETBYTV
D1AD: E0 10     298          CPX  #$10       ;is it 0-15?
D1AF: 90 03     299          BLT  HCOLOR0    ;Yes
D1B1: 4C 8C 03  300          JMP  IQERRV     ;No
D1B4: 8A        301 HCOLOR0  TXA
D1B5: 0A        302          ASL             ;Multiply by 4 for use with
D1B6: 0A        303          ASL             ;Color mask lookup table
D1B7: 85 E4     304          STA  HCOLORZ
D1B9: 60        305          RTS
                306
D1BA: 20 92 03  307 DCOLOR   JSR  ADDONV     ;Set hi/med res mode -  &COLOR
D1BD: 20 82 03  308          JSR  GETBYTV
D1C0: E0 02     309          CPX  #$02       ;legal arguments- 0 and 1
D1C2: 90 03     310          BLT  DCOLOR0    ;0 sets hires mode
D1C4: 4C 8C 03  311          JMP  IQERRV     ;1 sets medium res mode
D1C7: 8A        312 DCOLOR0  TXA
D1C8: F0 02     313          BEQ  DCOLOR1
D1CA: A9 80     314          LDA  #$80
D1CC: 85 E7     315 DCOLOR1  STA  RESMODE
D1CE: 60        316          RTS
                317
D1CF: 20 B8 03  318 DHFNS    JSR  FRMNUMV    ;HPLOT parsing subroutine-
D1D2: 20 C2 03  319          JSR  GETADRV    ;read x coor from program text
D1D5: A4 51     320          LDY  LINNUM+1   ;and convert to 2-byte num
D1D7: A6 50     321          LDX  LINNUM     ;in LINNUM
D1D9: C0 02     322          CPY  #>560      ;check range- legal 0-559
D1DB: 90 06     323          BLT  DHFNS1
D1DD: D0 C5     324          BNE  GGERR
D1DF: E0 30     325          CPX  #<560
D1E1: B0 C1     326          BGE  GGERR
D1E3: 8A        327 DHFNS1   TXA             ;hold onto it for a moment
D1E4: 48        328          PHA
D1E5: 98        329          TYA
D1E6: 48        330          PHA
D1E7: A9 2C     331          LDA  #','       ;verify comma between the coors
D1E9: 20 AE 03  332          JSR  SYNCHRV
D1EC: 20 82 03  333          JSR  GETBYTV    ;get the y coor
D1EF: E0 C0     334          CPX  #$C0       ;Less than 192?
D1F1: B0 B1     335          BGE  GGERR
D1F3: 86 9D     336          STX  DSCTMP     ;set registers for DHPOSN entry
D1F5: 68        337          PLA
D1F6: A8        338          TAY
D1F7: 68        339          PLA
D1F8: AA        340          TAX
D1F9: A5 9D     341          LDA  DSCTMP
D1FB: 60        342          RTS
                343
                344 * HIRES LEFT, RIGHT, UP, AND DOWN SUBROUTINES
                345
D1FC: 10 51     346 LFTRT    BPL  RIGHT      ;Use sign for left/right subr.
```

LISTING 2: AMPERDHR.D000 *(continued)*

```
D1FE: 24 E7      347             BIT   RESMODE     ;which mode are we in?
D200: 30 14      348             BMI   MEDLEFT    ;medium- do it differently
D202: C6 30      349 LEFT        DEC   HMASK
D204: 10 48      350             BPL   RTS2        ;Not <0 yet
D206: A0 06      351             LDY   #$06        ;wrap around for a lower "byte"
D208: 84 30      352             STY   HMASK
D20A: A4 E5      353             LDY   HNDX
D20C: 88         354             DEY
D20D: 10 02      355             BPL   LEFT2       ;<0 yet?
D20F: A0 4F      356             LDY   #$4F        ;yes, wrap around screen
D211: 84 E5      357 LEFT2       STY   HNDX
D213: 4C 31 D1   358             JMP   DHPOSN5     ;go get new HCOLOR for this "byte"
                 359
D216: C6 19      360 MEDLEFT     DEC   MSKPTR      ;for next pixel
D218: 30 0F      361             BMI   MEDLEFT2    ;less than zero- skip down
D21A: A6 19      362             LDX   MSKPTR      ;get new MSKPTR
D21C: BD 1C D5   363             LDA   MSKTBPTR,X  ;get new HMASK
D21F: C5 30      364             CMP   HMASK       ;is it the same as the old HMASK?
D221: D0 0D      365             BNE   MEDLEFT3    ;No, skip down
D223: BD 38 D5   366 LEFTOUT     LDA   PLOT2FLG,X  ;Yes, update PLOT2 flag
D226: 85 F9      367             STA   PLOT2
D228: 60         368             RTS
D229: A2 1B      369 MEDLEFT2    LDX   #$1B        ;It's a new group of four "bytes"
D22B: 86 19      370             STX   MSKPTR      ;wrap around the pointers
D22D: BD 1C D5   371             LDA   MSKTBPTR,X
D230: 85 30      372 MEDLEFT3    STA   HMASK
D232: A9 10      373             LDA   #$10        ;bit 4
D234: 24 F9      374             BIT   PLOT2       ;is bit 4 set on the PLOT2 flag?
D236: F0 EB      375             BEQ   LEFTOUT     ;no, don't update HNDX or HCOLOR
D238: C6 E5      376             DEC   HNDX        ;yes, we're into a new "byte"
D23A: 10 04      377             BPL   MEDLEFT4    ;so adjust pointers; is HNDX <0?
D23C: A9 4F      378             LDA   #$4F        ;yes, wrap around the screen
D23E: 85 E5      379             STA   HNDX
D240: BD 38 D5   380 MEDLEFT4    LDA   PLOT2FLG,X  ;no, update PLOT2 for next plot
D243: 85 F9      381             STA   PLOT2
D245: A5 E5      382 MEDLEFT5    LDA   HNDX
D247: 29 03      383             AND   #$03        ;Divide HNDX MOD 4
D249: 18         384             CLC
D24A: 65 E4      385             ADC   HCOLORZ     ;Calculate new color mask pointer
D24C: 85 1C      386             STA   HCOLOR1     ;(see DHPOSN)
D24E: 60         387 RTS2        RTS
                 388
D24F: 24 E7      389 RIGHT       BIT   RESMODE     ;mostly same as LEFT routines
D251: 30 1A      390             BMI   MEDRITE
D253: E6 30      391             INC   HMASK
D255: A5 30      392             LDA   HMASK
D257: C9 07      393             CMP   #$07
D259: 90 F3      394             BLT   RTS2        ;Not >6 yet
D25B: A0 00      395             LDY   #$00
D25D: 84 30      396             STY   HMASK
D25F: A4 E5      397             LDY   HNDX
D261: C8         398             INY
D262: C0 50      399             CPY   #$50
D264: 90 02      400             BLT   RIGHT2      ;HNDX not >79 yet
D266: A0 00      401             LDY   #$00        ;wrap around
D268: 84 E5      402 RIGHT2      STY   HNDX
D26A: 4C 31 D1   403             JMP   DHPOSN5     ;get new color mask
                 404
D26D: E6 19      405 MEDRITE     INC   MSKPTR      ;for medium res mode
D26F: A6 19      406             LDX   MSKPTR
D271: E0 1C      407             CPX   #$1C
D273: B0 0D      408             BGE   MEDRITE2
D275: BD 1C D5   409             LDA   MSKTBPTR,X
D278: C5 30      410             CMP   HMASK
D27A: D0 0D      411             BNE   MEDRITE3
D27C: BD 38 D5   412 RITEOUT     LDA   PLOT2FLG,X
D27F: 85 F9      413             STA   PLOT2
D281: 60         414             RTS
D282: A2 00      415 MEDRITE2    LDX   #$00
D284: 86 19      416             STX   MSKPTR
D286: BD 1C D5   417             LDA   MSKTBPTR,X
D289: 85 30      418 MEDRITE3    STA   HMASK
D28B: A9 0F      419             LDA   #$0F        ;bit 3
D28D: 24 F9      420             BIT   PLOT2       ;are we into a new "byte"?
D28F: F0 EB      421             BEQ   RITEOUT     ;No, don't update HNDX or HCOLOR
D291: E6 E5      422             INC   HNDX        ;yes
D293: A5 E5      423             LDA   HNDX
D295: C9 50      424             CMP   #$50
D297: 90 04      425             BLT   MEDRITE4
D299: A9 00      426             LDA   #$00
D29B: 85 E5      427             STA   HNDX
D29D: BD 38 D5   428 MEDRITE4    LDA   PLOT2FLG,X
D2A0: 85 F9      429             STA   PLOT2
D2A2: A5 E5      430             LDA   HNDX
D2A4: 29 03      431             AND   #$03
D2A6: 18         432             CLC
D2A7: 65 E4      433             ADC   HCOLORZ
D2A9: 85 1C      434             STA   HCOLOR1
D2AB: 60         435             RTS
                 436
D2AC: 10 16      437 UPDOWN      BPL   UP          ;Sign for up/down select.
D2AE: A6 E2      438 DOWN        LDX   Y0          ;load current y-coor
D2B0: E8         439             INX
D2B1: E0 C0      440             CPX   #$C0        ;over 191 yet?
D2B3: 90 02      441             BLT   DOWN1       ;no, skip down
D2B5: A2 00      442             LDX   #$00        ;yes, wrap around
D2B7: BD 87 D3   443 DOWN1       LDA   YLOOKUPH,X  ;get new base address and
D2BA: 85 27      444             STA   GBASH       ;store it in GBAS
D2BC: BD 47 D4   445             LDA   YLOOKUPL,X
D2BF: 85 26      446             STA   GBASL
D2C1: 86 F2      447             STX   Y0          ;save new y-coor
D2C3: 60         448             RTS
                 449
D2C4: A6 E2      450 UP          LDX   Y0
D2C6: CA         451             DEX
D2C7: E0 FF      452             CPX   #$FF        ;less than 0 yet?
D2C9: D0 02      453             BNE   UP1         ;no, skip down
D2CB: A2 BF      454             LDX   #$BF        ;yes, wrap at 191
D2CD: BD 87 D3   455 UP1         LDA   YLOOKUPH,X  ;get new base adress and store it
D2D0: 85 27      456             STA   GBASH
D2D2: BD 47 D4   457             LDA   YLOOKUPL,X
D2D5: 85 26      458             STA   GBASL
D2D7: 86 E2      459             STX   Y0          ;store new y-coor
D2D9: 60         460             RTS
                 461
```

```
                 462 * HIRES LINE DRAWING SUBROUTINES
                 463
D2DA: 48         464 DHLIN       PHA               ;On entry--
D2DB: 38         465             SEC               ;Accumulator = X0L
D2DC: E5 E0      466             SBC   X0L         ;X-reg = X0H
D2DE: 48         467             PHA               ;Y-reg = Y
D2DF: 8A         468             TXA
D2E0: E5 E1      469             SBC   X0H
D2E2: 85 D3      470             STA   QDRNT
D2E4: B0 0A      471             BCS   HLIN2
D2E6: 68         472             PLA
D2E7: 49 FF      473             EOR   #$FF
D2E9: 69 01      474             ADC   #1
D2EB: 48         475             PHA
D2EC: A9 00      476             LDA   #0
D2EE: E5 D3      477             SBC   QDRNT
D2F0: 85 D1      478 HLIN2       STA   DXH
D2F2: 85 D5      479             STA   EH
D2F4: 68         480             PLA
D2F5: 85 D0      481             STA   DXL
D2F7: 85 D4      482             STA   EL
D2F9: 68         483             PLA
D2FA: 85 E0      484             STA   X0L
D2FC: 86 E1      485             STX   X0H
D2FE: 98         486             TYA
D2FF: 18         487             CLC
D300: E5 E2      488             SBC   Y0
D302: 90 04      489             BCC   HLIN3
D304: 49 FF      490             EOR   #$FF
D306: 69 FE      491             ADC   #$FE
D308: 85 D2      492 HLIN3       STA   DY
D30A: 66 D3      493             ROR   QDRNT
D30C: 38         494             SEC
D30D: E5 D0      495             SBC   DXL
D30F: AA         496             TAX
D310: A9 FF      497             LDA   #$FF
D312: E5 D1      498             SBC   DXH
D314: 85 1D      499             STA   COUNTH
D316: B0 09      500             BCS   MOVEX2      ;Always taken
D318: 0A         501 MOVEX       ASL
D319: 86 9D      502             STX   DSCTMP
D31B: 20 FC D1   503             JSR   LFTRT
D31E: A6 9D      504             LDX   DSCTMP
D320: 38         505             SEC
D321: A5 D4      506 MOVEX2      LDA   EL          ;Carry is always set here
D323: 65 D2      507             ADC   DY
D325: 85 D4      508             STA   EL
D327: A5 D5      509             LDA   EH
D329: E9 00      510             SBC   #0
D32B: 85 D5      511 HCOUNT      STA   EH
D32D: 86 9D      512             STX   DSCTMP
D32F: 08         513             PHP
D330: 20 55 D1   514             JSR   DHPLOT1
D333: 28         515             PLP
D334: A6 9D      516             LDX   DSCTMP
D336: E8         517             INX
D337: D0 04      518             BNE   HLIN4
D339: E6 1D      519             INC   COUNTH
D33B: F0 49      520             BEQ   RTS4
D33D: A5 D3      521 HLIN4       LDA   QDRNT
D33F: B0 D7      522             BCS   MOVEX
D341: 86 9D      523             STX   DSCTMP
D343: 20 AC D2   524             JSR   UPDOWN
D346: A6 9D      525             LDX   DSCTMP
D348: 18         526             CLC
D349: A5 D4      527             LDA   EL
D34B: 65 D0      528             ADC   DXL
D34D: 85 D4      529             STA   EL
D34F: A5 D5      530             LDA   EH
D351: 65 D1      531             ADC   DXH
D353: 4C 2B D3   532             JMP   HCOUNT
                 533
D356: 20 72 D3   534 DLOAD       JSR   SETADRS     ;Move main hires page to aux page
D359: 38         535             SEC               ; &LOAD command
D35A: 8D 00 C0   536             STA   STOR80OF
D35D: 20 11 C3   537             JSR   AUXMOVE
D360: 8D 01 C0   538             STA   STORE80
D363: 60         539             RTS
                 540
D364: 20 72 D3   541 DSAVE       JSR   SETADRS     ;Move aux hires page to main page
D367: 18         542             CLC               ; &SAVE command
D368: 8D 00 C0   543             STA   STOR80OF
D36B: 20 11 C3   544             JSR   AUXMOVE
D36E: 8D 01 C0   545             STA   STORE80
D371: 60         546             RTS
                 547
D372: A9 00      548 SETADRS     LDA   #$00        ;Set address pointers for
D374: A2 20      549             LDX   #$20        ;memory move via AUXMOVE.
D376: 86 3D      550             STX   A1H
D378: 85 3C      551             STA   A1L
D37A: 86 43      552             STX   A4H
D37C: 85 42      553             STA   A4L
D37E: A9 FF      554             LDA   #$FF
D380: A2 3F      555             LDX   #$3F
D382: 86 3F      556             STX   A2H
D384: 85 3E      557             STA   A2L
D386: 60         558 RTS4        RTS
                 559
D387: 20 24 28   560 YLOOKUPH HEX  2024282C3034383C2024282C3034383C
D38A: 2C 30 34 38 3C 20 24 28
D392: 2C 30 34 38 3C
D397: 21 25 29   561             HEX  2125292D3135393D2125292D3135393D
D39A: 2D 31 35 39 3D 21 25 29
D3A2: 2D 31 35 39 3D
D3A7: 22 26 2A   562             HEX  22262A2E32363A3E22262A2E32363A3E
D3AA: 2E 32 36 3A 3E 22 26 2A
D3B2: 2E 32 36 3A 3E
D3B7: 23 27 2B   563             HEX  23272B2F33373B3F23272B2F33373B3F
D3BA: 2F 33 37 3B 3F 23 27 2B
D3C2: 2F 33 37 3B 3F
D3C7: 20 24 28   564             HEX  2024282C3034383C2024282C3034383C
D3CA: 2C 30 34 38 3C 20 24 28
D3D2: 2C 30 34 38 3C
D3D7: 21 25 29   565             HEX  2125292D3135393D2125292D3135393D
D3DA: 2D 31 35 39 3D 21 25 29
D3E2: 2D 31 35 39 3D
D3E7: 22 26 2A   566             HEX  22262A2E32353A3E22262A2E32363A3E
D3EA: 2E 32 36 3A 3E 22 26 2A
D3F2: 2E 32 36 3A 3E
```

```
D3F7: 23 27 2B  567           HEX  23272B2F33373B3F23272B2F33373B3F
D3FA: 2F 33 37 3B 3F 23 27 2B
D402: 2F 33 37 3B 3F
D407: 20 24 28  568           HEX  2024282C3034383C20242B2C3034383C
D40A: 2C 30 34 38 3C 20 24 28
D412: 2C 30 34 38 3C
D417: 21 25 29  569           HEX  2125292D3135393D2125292D3135393D
D41A: 2D 31 35 39 3D 21 25 29
D422: 2D 31 35 39 3D
D427: 22 26 2A  570           HEX  22262A2E32363A3E22262A2E32363A3E
D42A: 2E 32 36 3A 3E 22 26 2A
D432: 2E 32 36 3A 3E
D437: 23 27 2B  571           HEX  23272B2F33373B3F23272B2F33373B3F
D43A: 2F 33 37 3B 3F 23 27 2B
D442: 2F 33 37 3B 3F
                572
D447: 00 00 00  573  YLOOKUPL HEX  00000000000000008080808080808080
D44A: 00 00 00 00 00 80 80 80
D452: 80 80 80 80 80
D457: 00 00 00  574           HEX  00000000000000008080808080808080
D45A: 00 00 00 00 00 80 80 80
D462: 80 80 80 80 80
D467: 00 00 00  575           HEX  00000000000000008080808080808080
D46A: 00 00 00 00 00 80 80 80
D472: 80 80 80 80 80
D477: 00 00 00  576           HEX  00000000000000008080808080808080
D47A: 00 00 00 00 00 80 80 80
D482: 80 80 80 80 80
D487: 28 28 28  577           HEX  28282828282828A8A8A8A8A8A8A8A8
D48A: 28 28 28 28 28 A8 A8 A8
D492: A8 A8 A8 A8 A8
D497: 28 28 28  578           HEX  28282828282828A8A8A8A8A8A8A8A8
D49A: 28 28 28 28 28 A8 A8 A8
D4A2: A8 A8 A8 A8 A8
D4A7: 28 28 28  579           HEX  28282828282828A8A8A8A8A8A8A8A8
D4AA: 28 28 28 28 28 A8 A8 A8
D4B2: A8 A8 A8 A8 A8
D4B7: 28 28 28  580           HEX  28282828282828A8A8A8A8A8A8A8A8
D4BA: 28 28 28 28 28 A8 A8 A8
D4C2: A8 A8 A8 A8 A8
D4C7: 50 50 50  581           HEX  505050505050505D0D0D0D0D0D0D0D0
D4CA: 50 50 50 50 50 D0 D0 D0
D4D2: D0 D0 D0 D0 D0
D4D7: 50 50 50  582           HEX  505050505050505D0D0D0D0D0D0D0D0
D4DA: 50 50 50 50 50 D0 D0 D0
D4E2: D0 D0 D0 D0 D0
D4E7: 50 50 50  583           HEX  505050505050505D0D0D0D0D0D0D0D0
D4EA: 50 50 50 50 50 D0 D0 D0
D4F2: D0 D0 D0 D0 D0
D4F7: 50 50 50  584           HEX  505050505050505D0D0D0D0D0D0D0D0
D4FA: 50 50 50 50 50 D0 D0 D0
D502: D0 D0 D0 D0 D0
                585
D507: 01        586  MSKTABL  DFB  %00000001  ;DOUBLE HIRES MASKS
D508: 02        587           DFB  %00000010
D509: 04        588           DFB  %00000100
D50A: 08        589           DFB  %00001000
D50B: 10        590           DFB  %00010000
D50C: 20        591           DFB  %00100000
D50D: 40        592           DFB  %01000000
D50E: 0F        593           DFB  %00001111  ;AUX- MEDIUM RES MASKS
D50F: 00        594           DFB  %00000000  ;MAIN
D510: 70        595           DFB  %01110000  ;AUX
D511: 01        596           DFB  %00000001  ;MAIN
D512: 1E        597           DFB  %00011110  ;MAIN
D513: 00        598           DFB  %00000000  ;AUX
D514: 60        599           DFB  %01100000  ;MAIN
D515: 03        600           DFB  %00000011  ;AUX
D516: 3C        601           DFB  %00111100  ;AUX
D517: 00        602           DFB  %00000000  ;MAIN
D518: 40        603           DFB  %01000000  ;AUX
D519: 07        604           DFB  %00000111  ;MAIN
D51A: 78        605           DFB  %01111000  ;MAIN
D51B: 00        606           DFB  %00000000  ;AUX
                607
D51C: 07 07 07  608  MSKTBPTR HEX  07070707090909
D51F: 07 09 09 09
D523: 0A 0B 0B  609           HEX  0A0B0B0B0B0D0D
D526: 0B 0B 0D 0D
D52A: 0E 0E 0F  610           HEX  0E0E0F0F0F0F11
D52D: 0F 0F 0F 11
D531: 12 12 12  611           HEX  12121213131313
D534: 13 13 13 13
                612
D538: 10 00 00  613  PLOT2FLG HEX  100000008080BF ;Bits 6 and 7 tell the plot
D53B: 00 80 80 8F
D53F: 50 00 00  614           HEX  500000000080BF ;routine where to replot in
D542: 00 00 80 8F
D546: 50 40 00  615           HEX  504000000008F ;med res.  Bits 4 and 3 tell
D549: 00 00 00 8F
D54D: 50 40 40  616           HEX  504040000000F ;LEFT & RIGHT to update HNDX.
D550: 00 00 00 0F
                617
                618  * PLOT2FLG is used to indicate to the plotting routine
                619  * whether a second plot is necessary, and in which
                620  * neighboring screen "byte", for the medium-res mode.
                621  * These values are BITted.  If the N flag is set, the next
                622  * highest bit-mask and color mask are used on the next
                623  * highest byte.  If the V flag is set, the next lowest byte
                624  * is given similar treatment.  No flags set indicate no
                625  * more plotting is necessary.
                626
D554: 00 07 0E  627  BITTABL  HEX  00070E15   ;used for medium-res calculations
D557: 15

--End assembly--

1368 bytes

Errors: 0
```

END OF LISTING 2

```
        KEY PERFECT 4.0
            RUN ON
        AMPERDHR.D000
==========================
   CODE      ADDR# - ADDR#
 ------    -------------
   2979      D000 - D04F
   2450      D050 - D09F
   2D11      D0A0 - D0EF
   2965      D0F0 - D13F
   237E      D140 - D18F
   25F8      D190 - D1DF
   2A14      D1E0 - D22F
   2A63      D230 - D27F
   25B5      D280 - D2CF
   2632      D2D0 - D31F
   2A50      D320 - D36F
   28E9      D370 - D3BF
   28B9      D3C0 - D40F
   22D4      D410 - D45F
   27A0      D460 - D4AF
   2E01      D4B0 - D4FF
   2A2A      D500 - D54F
   0202      D550 - D557
 PROGRAM CHECK IS : 0558
```

LISTING 3: AMPERDHR.DEMO

```
10  REM  ************************
20  REM  *    AMPERDHR.DEMO     *
30  REM  *    BY STEVE MEUSE    *
40  REM  *  COPYRIGHT (C) 1983  *
50  REM  *    BY STEVE MEUSE    *
60  REM  ************************
70  HOME : VTAB 8: HTAB 10: PRINT "AMPERDHR D
    EMONSTRATION": PRINT : PRINT : PRINT "TH
    IS DEMONSTRATION REQUIRES AN APPLE //E";
    : PRINT "WITH AN EXTENDED 80-COLUMN CARD
    OR";: PRINT "AN APPLE //C."
80  PRINT : PRINT : PRINT "DO YOU HAVE ONE OF
    THE ABOVE?(Y/N)";: GET K$: IF K$ = "N" THEN
    END
90  PRINT : PRINT  CHR$ (4)"PR#3"
100  PRINT  CHR$ (4)"BLOAD AMPERDHR.SETUP": CALL
    790
110  TEXT : POKE  - 16298,0: HOME
120  &  COLOR= 0: LOMEM: 16385: GOTO 160: REM
    SET HI-RES MODE
130 K =  PEEK ( - 16384): IF K > 127 THEN 150
140  RETURN
150  POKE  - 16368,0: POP
160  TEXT : HOME : PRINT "      Double Hires D
    emonstrations"
170  PRINT : PRINT "  Your choices:"
180  PRINT " 1. Color bars": PRINT " 2. Doubl
    e Cross": PRINT " 3. Perspectives": PRINT
    " 4. Set Medium Resolution mode": PRINT
    " 5. Set High Resolution mode"
190  PRINT " 6. BSAVE Demonstration": PRINT "
    7. BLOAD Demonstration": PRINT " 8. Qui
    t"
200  PRINT : PRINT "Current resolution mode i
    s ";: IF  PEEK (231) > 127 THEN  PRINT "
    Medium.": GOTO 220
210  PRINT "High."
220  PRINT : PRINT "   Some of these patterns
    take a minute or two to develop.  Have
    fun watching!"
230  PRINT "Press any key to return to this m
    enu.": PRINT
240  VTAB 23: INPUT "Your choice, please? ";Q
    $:Q =  VAL (Q$): IF Q = 8 THEN  HOME : PRINT
    CHR$ (21): END
250  IF Q < 1 OR Q > 7 THEN 240
260  ON Q GOTO 270,270,270,730,750,770,850
270 D = 6:E = 6:C = 0:A = 1
280  HOME : &  HGR :DF = 1: POKE  - 16302,0: ON
    Q GOTO 300,380,540: GOTO 160
290  REM COLOR BARS
300 SC =  PEEK (231): &  COLOR= 1: FOR X = 0 TO
    526 STEP 35: &  HCOLOR= X / 35: FOR Y =
    X TO X + 32 STEP 4: &  HPLOT Y,0 TO Y,19
    1: NEXT : NEXT : POKE 231,SC
310 X = 1403: VTAB 21: POKE X,0: PRINT "BLACK
    ";: POKE X,20: PRINT "DARK GREEN";: POKE
    X,40: PRINT "DARK BLUE";: POKE X,60: PRINT
    "MEDIUM BLUE";
320  VTAB 22: POKE X,5: PRINT "MAGENTA";: POKE
    X,25: PRINT "GREY1";: POKE X,45: PRINT "
    VIOLET";: POKE X,65: PRINT "LIGHT BLUE";
330  VTAB 23: POKE X,10: PRINT "BROWN";: POKE
    X,30: PRINT "LIGHT GREEN";: POKE X,50: PRINT
    "GREY2";: POKE X,70: PRINT "AQUA";
340  VTAB 24: POKE X,15: PRINT "ORANGE";: POKE
    X,35: PRINT "YELLOW";: POKE X,55: PRINT
    "PINK";: POKE X,74: PRINT "WHITE";
350  POKE  - 16301,0
360  GOSUB 130: GOTO 360
370  REM DOUBLE CROSS
380 Y = 559
390  GOSUB 130: GOSUB 470: GOSUB 500
400  FOR X = 0 TO 559 STEP E:Y = Y - E: IF Y <
    0 THEN Y = 0
410  &  HPLOT Y,95 TO X 0: &  HPLOT Y,95 TO X
    ,189: NEXT
420 F = E *  - 1: FOR X = 559 TO 0 STEP F:Y =
    Y + E: IF Y > 559 THEN Y = 559
430  &  HPLOT Y,95 TO X,0: &  HPLOT Y,95 TO X
    ,189: NEXT
440  GOTO 380
450  GOSUB 130: GOTO 450
460  PRINT  CHR$ (7): GOTO 450
470  IF C = 1 THEN 490
480 C = 1: &  HCOLOR= 15:SP = 70: GOTO 520
490 C = 0: &  HCOLOR= 0:SP = 140: GOTO 520
500 D = D + A:E = E + A: IF D > 189 THEN D =
    189
510  IF E > 558 THEN 460
520  RETURN
530  REM PERSPECTIVES
540 MX = 280:MY = 96
550  &  HCOLOR= 15
560 P2 = 2 * 3.1415926535
570 D1 = P2 / 360
580 PI = 105
590 CX = 280:CY = 96
600  GOTO 620
610  FOR I = 1 TO 9000: NEXT I
620 DA = 360 *  RND (1) + 1: &  HCOLOR=  RND
    (1) * 15: &  CLEAR : &  HCOLOR= 15
630  FOR A = 0 TO 30 * P2 STEP DA * D1
640 RX = 280 + 140 * A / PI *  COS (A):RY = 9
    6 + 50 * A / PI *  SIN (A)
650  &  HCOLOR= 0
660  FOR G = 1 TO A / 22: &  HPLOT MX + G,MY +
    G TO RX + G,RY + G: NEXT : &  HCOLOR= 15
670  &  HCOLOR= 15
680  &  HPLOT MX,MY TO RX,RY:MX = RX:MY = RY
690  NEXT : GOSUB 130
700 MX = 280:MY = 96
710  GOTO 610
720  REM SET MEDIUM RES MODE
730  &  COLOR= 1: GOTO 160
740  REM SET HIGH RES MODE
750  &  COLOR= 0: GOTO 160
760  REM  BSAVE DEMO
770  IF DF = 0 THEN  HOME : VTAB 12: PRINT "U
    se options 1-3 to draw a picture": PRINT
    "before trying to BSAVE.": PRINT "Press
    any key to return to menu.": GET K$: PRINT
    : GOTO 160
780  POKE  - 16304,0: POKE  - 16302,0
790  PRINT  CHR$ (4)"BSAVE PIC,A$2000,L$2000"
    : REM  SAVE MAIN MEM IMAGE
800  &  SAVE : REM  TRANSFER AUX MEM TO MAIN
    MEM
810  PRINT  CHR$ (4)"BSAVE PIC.1X,A$2000,L$20
    00": REM  SAVE AUX MEM IMAGE
820  PRINT  CHR$ (4)"BLOAD PIC": REM  IF YOU
    WANT TO RESTORE THE PICTURE
830  GET K$: PRINT : GOTO 160
840  REM BLOAD.DEMO
850  HOME : VTAB 12: PRINT "Have you already
    created the files PIC": PRINT "and PIC.1
    X on this disk? (Y/N)";: GET K$: PRINT :
     IF K$ = "Y" OR K$ = "y" THEN 880
860  IF K$ = "N" OR K$ = "n" THEN 160
870  GOTO 850
880  &  HGR : POKE  - 16302,0
890  PRINT  CHR$ (4)"BLOAD PIC.1X": REM  LOA
    D AUX MEM IMAGE INTO MAIN MEM
900  &  LOAD : REM  TRANSFER IT TO AUX MEMORY
910  PRINT  CHR$ (4)"BLOAD PIC": REM LOAD MAI
    N MEMORY
920  GET K$: PRINT : GOTO 160
```

END OF LISTING 3

```
        KEY PERFECT 4.0
            RUN ON
        AMPERDHR.DEMO
=============================
   CODE       LINE# - LINE#
 ------      -------------
   B62F         10 -   100
   B3BA        110 -   200
   B709        210 -   300
   BB57        310 -   400
   6117        410 -   500
   3771        510 -   600
   6EB5        610 -   700
   93CC        710 -   800
   AAB2        810 -   900
   15DD        910 -   920
PROGRAM CHECK IS : 0B0B
```