
Artists'
Oil Colour
Crimson
Lake
146 S.I. Series 2
37ml
Artists'
Oil Colour
Ultramarine
37ml

FEATURE ARTICLE

AMPERPALETTE

In addition to double Hi-Res equivalents of the Hi-Res Applesoft commands, this package offers many other handy commands.

Double Hi-Res graphics provide Apple IIe, IIc and IIGS owners with a very high resolution display. Unfortunately, Applesoft contains no built-in routines to use this powerful display mode. DUBLSTUF is an industrial-strength machine language program that adds full, double Hi-Res programming capability to Applesoft. Through simple ampersand commands, your program can use the full 560-dot horizontal resolution in black-and-white or 140-dot resolution in a special unrestricted 16-color mode. DUBLSTUF also offers several new commands, including a super-fast move routine and a utility to swap any two Hi-Res pages in memory. DUBLSTUF isn't limited to double Hi-Res page 1 ($2000-$3FFF); it can access double Hi-Res page 2 ($4000-$5FFF) and even write on page 3 ($6000-$7FFF) and transfer the results for viewing to page 1 or 2. There's even a music routine that makes it easy to add tunes to your double Hi-Res display. To use DUBLSTUF, you must have at least a IIc or IIe with a revision B or later motherboard, and an extended 80-column card with its jumper installed. It isn't necessary to have the enhanced IIe, as the program doesn't use any of the 65C02 opcodes. DUBLSTUF is loaded at the bottom of DOS (ProDOS or DOS 3.3) and occupies 2K of memory (&8 pages). It's compatible with other ampersand (&) routines; any & command that isn't a DUBLSTUF command will be passed on to any & routine already in place.

INSTALLING DUBLSTUF

To use DUBLSTUF from a BASIC program, just BRUN it as follows before loading any Hi-Res pictures or specifying any string variables:

```
10 PRINT CHR$(4);"BRUN DUBLSTUF":D$=CHR$(4):PRINT
   D$"PR#3":PRINT:VTAB 21
```

DUBLSTUF will load itself into the Hi-Res page 1 area, move itself into high memory, fix the & vector to point to itself, and set HIMEM to protect itself from Applesoft. COLOR is set to 15 (white), SCALE to 1, and ROT to 0. The shape table area is set to begin at $99D3. You (or your program) must do a PR#3 to initialize the 80-column card before using any of the graphics commands.

The initialization process is a little different in Apple's two operating systems. If DOS 3.3 is running, DUBLSTUF does a MAXFILES1 command. In this condition, DOS 3.3 can't handle more than one file at a time; it can't even do a CATALOG until all text files are closed. If ProDOS is running, DUBLSTUF uses ProDOS routines to move its buffers down by eight (256-byte) pages and then installs itself in the vacant space. Memory space can get tight with ProDOS; DUBLSTUF can only use the main memory area up to $8000 for double Hi-Res storage since ProDOS reserves a minimum of 2K of buffer space below DUBLSTUF. If limited memory is a problem, DOS 3.3 is a better choice.

TABLE 1: Double Hi-Res Colors

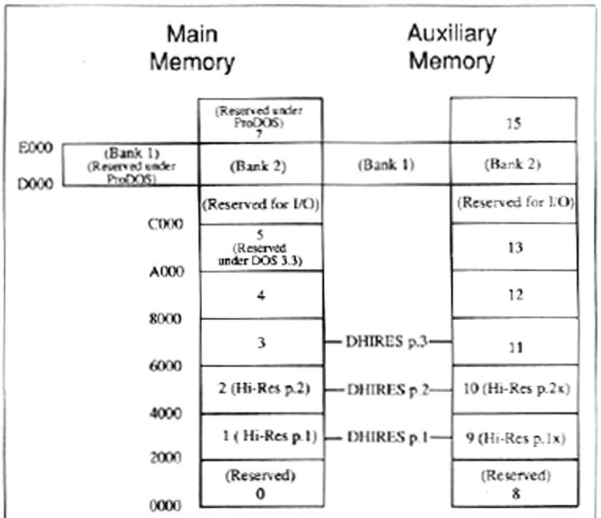
0 Black	8 Magenta
1 Dark blue	9 Purple
2 Dark green	10 Gray 2
3 Medium blue	11 Light blue
4 Brown	12 Orange
5 Gray 1	13 Pink
6 Green	14 Yellow
7 Aqua	15 White

ProDOS can access auxiliary memory by using it as a pseudodisk. You can convince ProDOS to cooperate with DUBLSTUF in the use of auxiliary memory by using the following program line near the beginning of your program:

```
20 PRINT: FOR X = 1 TO 7: PRINT CHR$(4);"BSAVE/RAM/
   PICTURE.";X;",A$2000,L$2000": NEXT
```

ProDOS will then think it has seven different Hi-Res pictures filed on its RAM disk, and it will be prevented from saving anything else in auxiliary memory.

FIGURE 1: 128K Apple Memory Map



** Figure shows a 128K Apple Memory Map. Numbers in areas refer to &M and &S blocks.*

DUBLSTUF COMMANDS

Many of the DUBLSTUF commands are identical to their Hi-Res counterparts, except that they're preceded by an &. Other commands are represented by a single letter following the &. These are the commands:

1. &HGR clears double Hi-Res page 1 to black, initializes it for drawing, and then displays it in mixed mode (text at bottom). It sets the 80-column mode and sets the color for plotting to white.
2. &HGR2 clears double Hi-Res page 2, then initializes and displays it in full-screen mode, with 80 columns and color like &HGR. This and other page 2 commands should be entered only from a program, not from the keyboard. Program commands that produce screen output such as INPUT, PRINT, and GET should not be used while viewing double Hi-Res page 2, because the 80STORE soft switch is turned off. Any of these commands will turn on 80STORE and cause double Hi-Res page 1 to be viewed instead. Waiting for a keypress using a WAIT 49152,128: POKE 49168,0 sequence will work fine.
3. COLOR= is the same as the Lo-Res color command. Please note that there's no & in front of this command. Color numbers and their associated double Hi-Res colors are shown in Table 1. Purists will note that these colors aren't in the same order as the double Hi-Res colors given in various Apple manuals. The actual color you see will depend on the color settings on your TV or monitor; RGB monitors may show colors different from the ordinary kind.
4. &HPLOT draws a 1-dot wide line on the drawing page; for example, &HPLOT 0,0 TO 559,191. The maximum horizontal value is 559, and the line may not appear if the selected color is not the same as the color of the dot being addressed.
5. &PLOT is the same as &HPLOT, but draws a 4-dot wide line. A plotted line will appear solid, but the effective horizontal resolution is only 140 dots per inch and plotting is slower than with &HPLOT. If the horizontal argument is above 556, some wraparound to the left side of the screen may occur. In general, use &HPLOT for horizontal lines and &PLOT for vertical ones. This sequence is a good example:

```
&HPLOT 0,0 TO 556,0: &PLOT TO 556,159: &HPLOT TO
0,159: &PLOT TO 0,0
```

6. &DRAW draws a shape in the specified color; for example, &DRAW 1 AT 100,100. When DUBLSTUF is BRUN, it initializes a small shape table at $99D3, consisting of a single star shape to test the &DRAW and &XDRAW commands. With ProDOS, this table must end at or before $99FF, but if you're using DOS 3.3, the shape area can be extended through $9AA5, for a maximum length of 210 bytes.
7. &XDRAW draws a shape in a color that's always the opposite of its background. &XDRAWing a shape and &XDRAWing it again will make the background reappear intact; otherwise, this command works just like &DRAW.
8. &INVERSE*x* makes the whole drawing screen invert itself — all off dots are turned on and all on dots are turned off. The drawing color is also inverted. The number following the command (*x*) indicates the screen to be inverted. It should be 1, 2, or 3. &INVERSE1: &INVERSE1 restores the original picture.
9. &CLEAR*x* clears the whole drawing screen to the last specified color. Again, the *x* parameter is used to specify screen 1, 2, or 3. COLOR=1: &CLEAR1 will produce a dark blue screen. The drawing color will be the opposite of the one on the screen.
10. &GR*x* lets you draw on the specified screen (1-3) without changing the current screen. If you are watching double Hi-Res page 1, for instance, &GR2 will let you plot on page 2, but you won't be able to see it until the program changes the viewing screen to page 2 or swaps the screens (see below).
11. &V*x* lets you view the specified screen page (1 or 2) in full mode. &V2 allows the viewer to see the plotting that's been done on page 2. The command doesn't affect the screen initialized for drawing. Screen 3 can't be viewed.
12. &Z*x* initializes the specified screen (1-3) for drawing and zeroes it to black. It also changes the drawing color to white.

After using the &Z, &GR, &CLEAR, &PRINT, or &INVERSE commands, the subsequent screen used for drawing will be the one specified in the command. The screen being viewed is not changed.

13. &F specifies full-graphics mode (same as a POKE 49234,0). This command doesn't change the screen being viewed.
14. &T lets you view double Hi-Res page 1 in mixed mode.
15. &W converts (widens) a Hi-Res picture to double Hi-Res by doubling the number of dots. This command works only for screen page 1, which must be cleared to black. Here's an example:

```
&HGR
BLOAD name of picture,A$2000
&W
```

The resulting picture on the double Hi-Res screen will look just like the original picture did on the Hi-Res screen, except for different colors. The color change is the result of converting the Hi-Res colors to double Hi-Res colors.

16. &M moves (duplicates) any 8K block of memory to any other 8K block. The Apple can do this with two built-in MOVE routines, but DUBLSTUF does it four times faster. The syntax must be as follows:

```
&M X TO Y
&M 2 TO 1
```

FIGURE 2: Double Hi-Res Sceen Bytes

	Byte 1							Byte 2							Byte 3						
Horizontal position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	2
Byte	$2000 - aux							$2000 - main							$2001 - aux						
Bit	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0	1	2	3	4	5	
Color bit	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	

In this case, a picture will be duplicated from Hi-Res page 2 to Hi-Res page 1. *Warning: Whatever was originally on page 1 will disappear.*

17. &S swaps any two 8K blocks of memory.

```
&S 2 TO 1
```

will make the contents of Hi-Res page 1 appear on page 2, and vice versa. It works more slowly than &M, but it's still as fast as the Monitor MOVE routines.

For the purposes of &M and &S only, the 128K of available memory is divided into sequentially numbered 8K blocks; blocks 0-7 reside in main memory and blocks 8-15 live in auxiliary memory (see **Figure 1**). There are some restrictions on the &M and &S commands:

a. Areas 0 and 8 are reserved. Areas 4 and 5 are reserved for DOS and DUBLSTUF. If ProDOS is in effect, areas 6 and 7 are also reserved. Attempting to use these areas will result in an ILLEGAL QUANTITY error.
b. Because of the complexity of using areas 6 and 14, you can't &M or &S *to* these areas. However, you can &M or &S *from* these areas freely. To save a picture in area 1 to area 14, for example, do &S 14 TO 1. You can then get it back with an &M 14 TO 1 or another &S 14 TO 1.
c. Moving or swapping between the two bank-switched memory areas can't be done: &S 15 TO 7 will have no effect. To use &M and &S to move or swap a double Hi-Res picture, use them twice. &M 1 TO 2: &M 9 TO 10 puts a duplicate of double Hi-Res page 1 on DHR page 2.

18. &P plays a note through the speaker. It should be followed by a variable for the duration, a comma (,), and a variable for the pitch (higher pitches = lower numbers). &P 500,260 plays a fairly long A tone. The middle octave notes of a piano are approximate by the following values:

A = 260	Bb = 246	B = 232	C = 219	C# = 207
D = 195	Eb = 184	E = 174	F = 164	F# = 155
G = 146	G# = 138			

To go down an octave, multiply these numbers by two. To go up an octave, divide by 2 (DUBLSTUF rounds it down to the nearest integer). High notes tend to sound flat; for the best sound quality, each note should be tuned, and notes with a value of less than 65 should be avoided completely. The IIc will play slightly higher and faster notes than the IIe. This routine has several features:

a. It can handle numbers up to 32767 (a very low or very long note).
b. Except on the IIc, which doesn't have cassette connections, it sends the same note out the "cassette out" jack. This jack can be connected to an amplified speaker for much better sound quality. (Use the earphone jack on the IIc without an amplifier.)
c. A pitch of 0 produces a rest, which is also useful for program pauses.

19. &C accesses the Monitor MOVE (copy) routine for main memory-to-main memory moves. If DOS 3.3 is in use, it can be used to move data into, but not out of, the main 16K area. Syntax for the command is as follows:

```
&C from start,from end TO destination start
```

with the names in italics being replaced by decimal memory locations. For instance, to move a 256-byte shape table loaded at $6000 (decimal 24576) to location $8000 (decimal 32768), use this code:

```
30 &C 24576,24831 TO 32768
```

20. &A accesses the AUXMOVE routine to move memory from main to auxiliary or vice versa. It has the same syntax as the &C command, except that the destination address must be followed by a comma, and an M to move auxiliary to Main, or an A to move main to Auxiliary. To save your shape table in auxiliary memory, type:

```
40 &A 24576,24831 TO 32768,A
```

&A can move data in and out of the main 16K bank if DOS 3.3 is in use. With both &C and &A, specifying an address in the $Cxxx space (decimal 49152 to 53247) will result in $Dxxx bank 1 being used instead. The auxiliary 16K bank can't be accessed. Since these routines are much slower than &M, there's no reason to use them for moving 8K pages.

21. &WAIT waits for the start of the vertical blanking interval. This command has no effect on the IIc. DUBLSTUF uses &WAIT before showing a double Hi-Res screen, or to do an &DRAW or &XDRAW. This minimizes video glitches, which occur when the video raster sweeps through an area of the screen that's being changed; the delay averages 1/120th second and is never longer than 1/60th second.

22. &PRINTx does a double Hi-Res screen dump to your printer, and is followed by one or more page numbers (1, 2, or 3) separated by commas. Depending on your printer, the picture will be approximately 8 inches wide by 5 1/2 inches high. This command should be followed by a PRINT statement to reset the printer and let subsequent operating system commands function properly. Printing will be in non-inverse format (white screen to white paper) unless an INVERSE command is in effect. To dump an inverse version of page 1 immediately followed by page 2 without any intervening space, use:

```
70 INVERSE : &PRINT 1,2 : NORMAL : PRINT
```

This command is complicated, since different printers require different control codes. I've included code to use either the Apple dot-matrix series of printers or Epson printers (see the Entering the Program section). The plethora of different printer cards also makes standardizing difficult. Your printer card must be initialized so that it will not echo characters to the screen or send any extraneous carriage returns (CRs) to the printer. If you have a Grappler+ card in slot 1, the proper codes to print page 1 would be:

```
50 PRINT D$"PR#1" : REM No Escape Control-Q
   necessary
60 PRINT CHR$(9)"0N" : REM No CRs, no echo
70 &PRINT 1: PRINT
80 PRINT CHR$(9)"e" : REM Printer off,
   screen on
```

If you're using a IIc or IIGS with a printer connected to port 1, use the following codes:

```
50 PRINT D$"PR#1"
60 PRINT CHR$(9)"Z"
80 PRINT D$"PR#3" : PRINT : VTAB 21
```

A Super Serial Card will respond to the following codes:

```
50 PRINT CHR$(27) CHR$(17): REM Escape Control-Q
   = 80 card off
60 PRINT D$"PR#1" : REM No CRs, no echo is
   default
80 PRINT D$"PR#3" : PRINT : VTAB 21
```

Some cards, such as the old Apple parallel printer card, do not have a command to suppress carriage returns. DUBLSTUF recognizes an Apple parallel printer card in slot 1 and will reset its column count after every 14 characters. Try the following initialization codes:

```
50 PRINT CHR$(27) CHR$(17)
60 PRINT D$"PR#1" : PRINT CHR$(9)"255N"
80 PRINT CHR$(9)"80N" : PRINT D$"PR#3" : PRINT
   : VTAB 21
```

You will get faster throughput of both text and graphics to your printer if you set the printer and the card so that the printer supplies its own linefeed after a carriage return, rather than having the printer card do it. It's unnecessary to worry about codes

having to do with the "high bit" or "bit 7" because DUBLSTUF doesn't use this bit.

Using Double Hi-Res Graphics

There are only a few differences between Hi-Res and double Hi-Res graphics. Main memory requirements are the same. The maximum double Hi-Res coordinates are 559 horizontal and 191 vertical. While Hi-Res has only six different colors with restrictions on mixing colors in the same byte, double Hi-Res has 16 colors with no restrictions. Since there are twice as many horizontal dots to be displayed on the same monitor screen, the dots are closer together. On the average monitor, a small square shape will have about 5 horizontal dots for each 2 vertical dots. Therefore, in equations producing squares, circles and other geometric figures, you should multiply the horizontal axis by 2.5 in relation to the vertical one. Shapes drawn from Hi-Res shape tables will also be squeezed horizontally. If this is undesirable, then a new shape table with more horizontal plotting than vertical will need to be made.

LOADING AND SAVING DOUBLE HI-RES

Loading and saving Double Hi-Res pictures is complicated, since only the Hi-Res 1x area of auxiliary memory is available to DOS, and only while it is being viewed, *and* if the program includes a POKE 49237,0 to access the auxiliary portion of the memory and a POKE 49236,0 to access the main portion of the memory. Saving a Double Hi-Res picture as one continuous block is a better method. It shortens disk drive run time and usually uses less disk space. The following program line illustrates this technique:

```
40 &M 9 TO 2: POKE 49236,0: PRINT D$;"BSAVE name
   of picture,A$2000,L$3FFB"
```

POKE 49236,0 ensures main $2000 access. Of course, if you use the &M routine to move pictures, you must be sure you're not overwriting some other picture that you want to save.

To load the picture, just reverse the process:

```
50 POKE 49236,0: PRINT D$;"BLOAD name of picture,
   A$2000":&M 2 TO 9
```

If you're loading more than one Double Hi-Res picture, just move each one of them into two unused areas of auxiliary memory before loading the next one.

If you've used the /RAM feature of ProDOS as suggested previously, the auxiliary memory portions of your pictures will be filed as /RAM/PICTURE.*X*, where *X* has the value 1 for double Hi-Res page 1, 2 for page 2, and 3 for page 3. Pictures 4, 5, 6, and 7 will be whatever you've saved in the rest of the auxiliary areas.

ENTERING THE PROGRAM

If you have an assembler that can handle multiple ORGs, enter the source code from **Listing 1**, save it and assemble it to the object file DUBLSTUF. If you don't have an assembler that can handle multiple ORGs, enter the Monitor with CALL −151 and key in the hex code from **Listing 2**. Save the program with the command:

BSAVE DUBLSTUF,A$318A,L$/86F

Then enter **Listing 3** and save it with the command:

SAVE DUBL.DEMO

For help with entering *Nibble* listings, see the Typing Tips section.

HOW IT WORKS

If you're not familiar with the concept of double Hi-Res graphics you can find an explanation of it in several Apple manuals, such as *The Apple IIe Technical Reference Manual* and *The Apple IIc Reference Manual*. However, these manuals only hint at the possibilities of using double Hi-Res page 2. Several articles on double Hi-Res programmig have also appeared in *Nibble*.

Double Hi-Res is to Hi-Res as 80-column output is to 40-column output; there are twice as many dot positions horizontally. The regular dots occupy 8K of memory in main RAM, while the extra dots are found at the same memory address, but in auxiliary memory (built into the IIc, or on the extended 80-column card for the IIe). The bytes are interleaved, with each auxiliary memory byte displayed to the left of its main memory twin (see **Figure 2**). Only seven bits in each byte are used for color information. (The eighth bit is ignored in double Hi-Res, whereas in regular Hi-Res it determines whether to shift the image or not.) Pixels appear in the opposite order on the screen — i.e., the first color pixel is coded by bits 0-3 of the first byte.

In double Hi-Res, the number of bits of graphic information is doubled to 560 per line. In monochrome graphics, that means 560 pixels across. However, since 16 colors are available and four bits are required to code all the possibilities, the effective color resolution is 140 pixels across. In **Figure 2**, the color group numbers refer to the order of the pixels on the screen.

To make its way through this complex system, DUBLSTUF has to keep track of several variables. As with the Applesoft Hi-Res routines, locations $26-$27 hold the vertical screen position; the Y-Register and $E5 hold the horizontal offset; and $1C is the running color mask. Hi-Res uses location $30 as a byte mask. DUBLSTUF uses $FD instead of $30 to hold the color, which is produced by the Monitor Lo-Res color routine. In addition, the contents of location $FC tell DUBLSTUF which memory bank to address.

There are two ways to access the auxiliary memory. The easy way is to use the PAGE2 switches with the 80STORE switch on. 80STORE is normally on in 80-column mode. When it's on, an STA $C055 (POKE 49237,0) accesses auxiliary memory locations $2000-$3FFF for reading and writing, an STA $C054 (POKE 49236,0) enables main memory, and only Hi-Res page 1 can be used. If 80STORE is off, with an STA $C000 (POKE 49152,0), an STA $C055 displays double Hi-Res page 2 (or Hi-Res page 2, if double Hi-Res is turned off). An STA $C054 displays double Hi-Res page 1, but only main memory is available for reading and writing.

The daring way to use auxiliary memory is to turn the troublesome 80STORE switch off, and let the RAMRD and RAMWRT switches control switching between main and auxiliary memory. When RAMWRT is on (STA $C005), the microprocessor writes to auxiliary memory in the range $200-$BFFF. When RAMRD is on (STA $C003), it reads auxiliary memory $200-$BFFF. The problem with this approach is that a program residing in main memory that does an STA $C003 will switch itself out and crash. To use the switches, the portion of the program doing the memory reading and switching must reside somewhere else, such as in page 0 ($0000-00FF) or above $CFFF, since these areas are not affected by RAMRD and RAMWRT.

DUBLSTUF isn't limited to double Hi-Res page 1; it can even write on page 3.

DUBLSTUF solves this problem by swapping a short subroutine into page 0, doing a JSR to the subroutine that does the reading and writing of auxiliary memory, and then swapping the subroutine back into its normal program location. The program is highly self-modifying and a crash is likely to leave RAM memory in shreds, with a portion of the program replaced by a bunch of page 0 values and vice versa. The memory swapping also slows down the program a little. The big advantage of the method, though, is that all of auxiliary memory can be addressed, not just the page 1 area, and double Hi-Res page 2 can be displayed. When DUBLSTUF is finished plotting, the 80STORE switch is restored to its original state.

MODIFICATIONS

Epson Printer

If you have an Epson MX-80 printer with Graftrax or another Epson that will respond to the same codes, make the following substitutions near the end of DUBLSTUF:

```
1137 PINS DFC 3,$C,$30 ;Epson MX80 etc
1138 INIT DFC $1B,$41,$86,0,0,0,0,0
         printer
1139 LINE DFC $D,$20,$20,$20,$20,$1B,$4C,$48,$83
1140      DFC $1B,$C0,0,0,0
```

If your printer isn't compatible with either the Epson or Apple dot-matrix, you're pretty much on your own to find the right codes, but here are some guidelines:

1. **Line 1137** contains the pin codes to activate the bottom two pins of the printer, than the next two pins, and the next two pins. The pin representing bit 7 should not be used. If the first pin's code is 3, DUBLSTUF will assume an Epson is present and will send three characters for every two dot positions. If this creates a conflict, try codes 6,$18,$60.
2. **Line 1138** contains code to initialize the printer to 6/72 inch spacing. If possible, horizontal spacing should be 1/72 inch between dots and the printer should print unidirectionally. All codes must have the high bit clear (0) except the last one, which must have the high bit set (1). The line must contain eight codes, so fill out any unused spaces with zeros.
3. **Line 1139** contains the following: a mandatory carriage return, enough spaces to center the picture, and a code to say "graphics coming" (560 graphics characters for the Apple Dot Matrix, 840 for the Epson), with high bits cleared or set as per line 1138.
4. **Line 1140** contains the code to reset the printer to its normal state, with high bits cleared or set as per line 1138. Lines 1139 and 1140 must have a total of 14 bytes, so fill out any unused spaces with zeros.

More Space Below

If you're using ProDOS, you can use the following routine to get more space below DUBLSTUF:

```
LDA #X ;X = number of pages of memory needed
JSR BEF5
```

The routine returns with the Carry cleared, if successful, and the address of the new beginning of the ProDOS buffers in the Accumulator. If you're using DOS 3.3, you can get more space below DUBLSTUF simply by lowering HIMEM.

LISTING 1: DUBLSTUF Source Code

```
 0 :
 1 ************************
 2 * DUBLSTUF             *
 3 * Double Hi-Res Routines *
 4 * by DAVID L SMITH MD  *
 5 * Copyright (C) 1987   *
 6 * by MicroSPARC Inc.   *
 7 * Concord, MA  01742   *
 8 ************************
 9 * MicroSPARC Assembler 3.0
10
11 * EQUATES:
12 ZEROPG   EQU $0000      ;Plotting subroutine loc
13 SHAPE    EQU $1A        ;Working shape vect/scratch
14 HCOLOR1  EQU $1C        ;Running color mask
15 GBAS     EQU $26        ;Vertical screen loc
16 COLOR    EQU $30        ;Lo-res/DHIRES color
17 INVFLG   EQU $32        ;INVERSE/NORMAL mask
18 A1       EQU $3C        ;MOVE start address
19 A2       EQU $3E        ;MOVE end address
20 A4       EQU $42        ;MOVE dest address
21 LINNUM   EQU $50        ;Stash for 2-byte integers
22 CHRGET   EQU $00B1      ;Get next A'soft token
23 CHRGOT   EQU $00B7      ;Get last A'soft token
24 TO       EQU $C1        ;A'soft 'TO' token
25 YSAV     EQU $E5        ;Temp save for horiz index (Y)
26 HPAG     EQU $E6        ;Hi-Res writing page
27 SCALE    EQU $E7
28 SHAPEV   EQU $E8        ;Shape table vector
29 COLLIDE  EQU $EA        ;Collision counter
30 ROT      EQU $F9
31 MODE     EQU $FB        ;DRAWvsXDRAW, PLOTvsHPLOT, etc
32 LOC      EQU $FC        ;Aux or main memory
33 HMASK    EQU $FD        ;Running bit index
34 S80STATE EQU $FE        ;Save status of 80STORE switch
35 P3AMPERV EQU $3F5       ;On & go here
36
37 * SOFT SWITCHES:
38 C80STORE EQU $C000      ;80STORE off
39 S80STORE EQU $C001      ;80STORE on
40 RMAINRAM EQU $C002      ;Read main mem
41 RCARDRAM EQU $C003      ;Read aux mem
42 WMAINRAM EQU $C004      ;Write main mem
43 WCARDRAM EQU $C005      ;Write aux mem
44 SETSTDZP EQU $C008      ;Main z pg, stak, bnk sw mem
45 SETALTZP EQU $C009      ;Aux of above
46 SET80VID EQU $C00D      ;80-column mode
47 R80STORE EQU $C018      ;Rd 80STORE status (Neg = on)
48 RDVBLBAR EQU $C019      ;Rd video blank status (//e)
49 RD80COL  EQU $C01F      ;Rd 80COL switch (Neg = on)
50 TAPEOUT  EQU $C020      ;Cassette out tog (not on //c)
51 SPKR     EQU $C030      ;Speaker output toggle
52 TXTCLR   EQU $C050      ;Graphics mode
53 TXTSET   EQU $C051      ;Text mode
54 MIXCLR   EQU $C052      ;Full graphics
55 MIXSET   EQU $C053      ;Mixed graphics/text
56 TXTPAGE1 EQU $C054      ;HIRES p. 1 (or main mem)
57 TXTPAGE2 EQU $C055      ;HIRES p. 2 (or aux mem)
58 HIRES    EQU $C057      ;HIRES on
59 STDHIRES EQU $C05E      ;DHIRES on (if IOUDIS on)
60 STIOUDIS EQU $C07E      ;Enable DHIRES on
61 ROMR1#2  EQU $C081      ;Read ROM, write RAM bnk2
62 RIMBNK2  EQU $C083      ;Read/write RAM bank 2
63 RIMBNK1  EQU $C08B      ;R/W RAM1
64
65 * DOS/ROM GOODIES:
66 MAXFILES EQU $A258      ;DOS 3.3 MAXFILES
67 GETBUFR  EQU $BEF5      ;Get space above ProDOS buffers
68 FREEBUFR EQU $BEF8      ;Eliminate space above buffers
69 MACHID   EQU $BF98      ;ProDOS machine ID byte
70 KERNEL   EQU $BFFF      ;ProDOS Kernel # (Neg=DOS 3.3)
71 MOVEAUX  EQU $C311      ;Move files aux <-> main
72 OOMERR   EQU $D410      ;?OUT OF MEMORY ERROR
73 DATA     EQU $D995      ;Back into A'soft from &
74 FRMNUM   EQU $DD67      ;Eval formula into FAC $9D A3
75 CHKCOM   EQU $DEBE      ;LDA Comma, fall into SYNCHR
76 SYNCHR   EQU $DEC0      ;Ck syntax:bad->ERR else CHRGET
77 AYPOSINT EQU $E10B      ;INT(FAC) -> $A1-lo, $A0-hi
78 IQERR    EQU $E199      ;?ILLEGAL QUANTITY ERROR
79 GETBYT   EQU $E6F8      ;Value of txptr into X
80 GETADR   EQU $E752      ;INT(FAC) -> LINNUM
81 INTY     EQU $F4D3      ;Incr or decr vertical
82 INCRY    EQU $F504      ;Incr vertical
83 STASH    EQU $F5B2      ;Bit mask table (bit 7 on)
84 IDBYT0   EQU $FBB3      ;= $6 if //e or //c
85 IDBYT1   EQU $FBC0      ;=0 if //c (no cassette or VBL)
86 COUT     EQU $FDED      ;Send a char to output device
87 MOVE     EQU $FE2C      ;Move main to main if Y = 0
88 ROMRTS   EQU $FF58      ;Known RTS (normal & vector)
89
90          ORG $318A
91          LDA KERNEL     ;Move DUBLSTUF to high memory
92          BMI DOS3.3     ; and initialize it
93          LDA MACHID
94          AND #$10       ;Check memory size
95          BEQ PRTOOM
96          JSR FREEBUFR   ;Ensure ProDOS normal state
97          LDA #8         ;# free pages needed
98          JSR GETBUFR    ;Make 2K of free space
99          BCC LIFT       ;If no error
100 PRTOOM  JMP OOMERR     ;Not a 128K machine
101 DOS3.3  LDA IDBYT0
102         CMP #6
103         BNE PRTOOM
104         LDA #1
105         JSR MAXFILES   ;MAXFILES1
106         LDA #START
107         LDY /START
108         STA LINNUM
109         STY LINNUM+1
110         JSR $F28E      ;Into HIMEM routine
111 LIFT    LDA #$20       ;HIRES write to p. 1
112         STA HPAG
113         LDY #$FF
114         STY COLOR
115         STY A2         ;Source end lo byte
116         INY            ;Y = 0
117         STY A1         ;Source start lo byt
118         STY A4         ;Dest start lo
119         STY ROT
120         LDA /$32
121         STA A1+1       ;Source start hi
122         LDA #$39
123         STA A2+1       ;Source end hi
124         LDA #START/
125         STA A4+1       ;Dest start hi
126         JSR MOVE       ;Call the van
127         LDA #SHTBL
128         STA SHAPEV     ;Init shape tbl
129         LDA #SHTBL/
130         STA SHAPEV+1
131         INY
132         STY SCALE
133         LDA P3AMPERV+2 ;Get old & vector
134         CMP #START/    ;Already initialized?
135         BEQ RTS0       ;If so, quit
136         STA NUAMPERV+2 ;Piggyback it on ours
```

LISTING 1: DUBLSTUF Source Code *(continued)*

```
137           LDA P3AMPERV+1
138           STA NUAMPERV+1
139           LDA #START
140           STA P3AMPERV+1    & -> our parser
141           LDA #START/
142           STA P3AMPERV+2
143 RTS0      RTS
144
145           ORG $9200
146 START     BIT ROMRIM2       ;May need to RAM write later
147           LDX #TABLE1-TABLE0
148 PARSER    CMP TABLE0-1,X    ;Match byte in A
149           BEQ PUSH
150           DEX
151           BNE PARSER
152 NUAMPERV  JMP ROMRTS        ;Not ours  try old &v
153 PUSH      PHP               ;Save interrupt status
154           LDA #QUIT/        ;Make routine RTS goto QUIT
155           PHA
156           LDA #QUIT-1
157           PHA
158           TXA
159           ASL
160           TAX
161           LDA TABLE1-1,X    Routine address hi byte
162           PHA
163           DEX
164           LDA TABLE1-1,X    Routine address lo byte  1
165           PHA               ;Addresses saved on stack
166           LDA R80STORE      ;Get 80STORE state
167           STA S80STATE
168           BPL +5            ;If off
169           STA TXTPAGE1      ;Continue showing DHIRES p.1
170           JMP CHRGET        ;Then goto address on stack +1
171
172 QUIT      PLP               ;Get old 'rupt status
173           JSR CHRGOT        ;Make acc = prog byte
174           JMP DATA          ;Continue program execution
175
176 TABLE0     ASC 'ACEMPSTVWZ' ;Letter commands
177           DFC $88,$80,$90,$91,$9J,$94 ;Token commands
178           DFC $95,$9E,$85,$8A,$8D
179 TABLE1     ADR A-1,C-1,F-1,M-1,P-1,S-1,T-1,V-1,W-1,Z-1
180           ADR GR-1,PLOT-1,HGR2-1,HGR-1,HPLOT-1,DRAW-1
181           ADR XDRAW-1,INVERSE-1,HAIT-1,PRINT-1,CLEAR-1
182 GETINT    JSR FRMNUM        ;Eval prog byt & ff, as formula
183           JMP GETADR        ;Integer it into 2 bytes
184
185 F         STA MIXCLR        ;Full graphics mode
186 RTS1      CLI               ;Restore interrupts
187           RTS
188
189 P         JSR GETINT        ;Play a note
190           JSR CHKCOM        ;Duration into LINNUM
191           JSR FRMNUM        ;Comma and pitch
192           JSR AYPOSINT      ;Reduce to 2 bytes
193           INC $A4           ;A4 = 1
194           LDA $A1           ;Get pitch lo byte  0 if a rest
195           BNE CKID          ;Not a rest
196           DEC $A0           ;Pitch hi byte -> $FF if rest
197 CKID      LDA IO@YT1        ;Hold //c status in A
198           SEI               ;Don't shoot the piano player
199 GETPITCH  LDX $A1           ;Hold pitch lo in X
200           LDY $A0           ;Hold pitch hi in Y
201           BMI DURCNT        ;If a rest
202           BIT SPKR          ;CLICK!
203           CMP #0            ;//c?
204           BEQ DURCNT
205           BIT TAPEOUT       ;(click)
206 DURCNT    DEC $A4           ;Duration register 0
207           BNE PITCH
208           DEC LINNUM        ;Duration register 1
209           BNE PITCH
210           DEC LINNUM+1      ;Duration register 2
211           BMI RTS1
212 PITCH     DEX
213           BNE DURCNT
214           DEY
215           BMI GETPITCH      ;Reload registers, whap spkr
216           BPL DURCNT        ;Always
217
218 P@SWAP     DEX               ;Swap a routine into p.@
219           STX P@UNSWAP+1    ;Modify routine below to
220           STY LOOP@+1       ;  use direct addressing
221           STY SAV@+1
222           STA LOOP@+2
223           STA SAV@+2
224 P@UNSWAP  LDX #@            ;Restore routine and p.@
225 LOOP@     LDA $FFFF,X       ;  when you are done
226           LDY $@,X
227           STA $@,X
228           TYA
229 SAV@      STA $FFFF,X
230           DEX
231           BPL LOOP@
232           INX               ;X = @
233           RTS
234
235 INIT1     LDA HPAG          ;Initialize some routines
236 INIT2     STA SMAPE+1
237           LDY #@
238           STY SMAPE
239           STA C@STORE       ;Turn it off
240           RTS
241
242 * Routine moved to p. @ to do actual work of M(ove)
243 MOVER     STA RMAINRAM      ;Altered to $C005 if to alt mem
244           STA RMAINRAM      ;Changed to $C003 if from alt
245 LOOP4     LDA $6000,X
246 SAV1      STA $4000,X
247           INX
248           BNE LOOP4
249           LDA $@8
250           CMP #$5F          ;Moved all bytes yet?
251           INC $@8
252           INC $@B           ;Self-modifying code
253           BCC LOOP4
254           STA RMAINRAM      ;To go back to CARRIER
255           RTS
256
257 SWAPPER   STA RMAINRAM      ;Similar routine for S(wap)
258           LDA $6000,X
259 SW@       STA RMAINRAM
260 LD1       LDY $4000,X
261 SW1       STA WMAINRAM
262 SAV2      STA $4000,X
263           STA WMAINRAM
264           TYA
265           STA $6000,X
266           INX
267           BNE SWAPPER
268           LDA $@5
269           CMP #$5F
270           INC $@5
271           INC $@B
272           INC $11
273           INC $18
274           BCC SWAPPER
275           STA RMAINRAM
276           RTS
277
278 GETLOC    JSR GETINT        ;Get an address in LINNUM
279           CMP #$C@          ;Check for bankswitched
280           BCC RTS6          ;No
281           LDX KERNEL        ;If bankswitched, is ProDOS?
282           BPL ERR           ;Yes, bankswitched in use
283           CMP #$D@          ;Is it in $Cx area?
284           BCS RTS6          ;No, so quit
285           LDX #8            ;Yes, so map to $Dxxx bank 1
286           STX LOC           ;X is index for flip-flop
287           ORA #$10          ;$Cx -> $Dx
288 RTS6      RTS
289
290 XCHEQUE   LDX #4            ;Make sure X in allowed range
291 XCHEQUE1  STX $9D           ;Temp stash upper bound
292 XCHEQUE2  JSR GETBYT        ;Val into X
293           TXA               ;Must be >@
294           BEQ ERR
295           CMP $9D
296           BCS ERR
297           RTS
298
299 SETUP     ASL               ;Process & err chkr for M & S
300           ASL               ;Make a pseudo-addr (addr/2)
301           ASL
302           ASL
303           CMP #$40          ;Hi-Res areas 1,2,3 OK
304           BCC RTS2
305           CMP #$60          ;Don't mess with DOS area
306           BCC ERR
307           CMP #$80          ;$@@@@ area in alt ram not OK
308           BEQ ERR
309           CMP #$90          ;$2@@@ in alt ram
310           LDX KERNEL        ;Check for ProDOS
311           BMI CKALTZP       ;No, must be 3.3
312           BCC ERR           ;ProDOS preempts main 16K space
313 CKALTZP   CMP #$E@          ;Sets carry if alt 16K bank
314 RTS2      RTS
315
316 ERR       STA SETSTDZP      ;Illegal quantity error handler
317           LDA RD80COL       ;In 80-col. mode?
318           BPL +5
319           STA S80STORE      ;If so, turn on 80STORE
320           STA TXTPAGE1      ;Make sure looking at p.1
321           STA TXTSET        ;To read error message
322           JMP IQERR
323
324 M         LDX #@            ;Move an 8K area
325 S         STX MODE          ;Swap 8K areas
326           LDX #$10          ;Make sure in range
327           JSR XCHEQUE1
328           STA COLLIDE       ;Temp storage
329           LDA #T@           ;Make sure next char is 'T@'
330           JSR SYNCHR        ;Check syntax and get new char
331           JSR XCHEQUE2      ;Check range of 2nd address
332           CMP #6            ;"T@" $D@ area not allowed
333           BEQ ERR
334           CMP #$E           ;Alt 16K $D@ area
335           BEQ ERR
336           STA C8@STORE
337           JSR SETUP         ;Process and check it
338           LDY COLLIDE       ;Get 1st address back
339           LDX MODE          ;Move or swap?
340           SEI               ;NO 'RUPTS IF ROMS DISABLED
341           BCC MAINZ         ;Not alt mem
342           STA SETALTZP      ;In alt mem
343 MAINZ     BNE SWAP
344 MOVE@     ASL               ;Make real address. Alt ->SEC
345           STA SAV1+2
346           LDA #4            ; use fast direct addressing
347           ADC #@            ;Adds 1 if carry set
348           STA MOVER+1       ;To set proper switch
349           TYA               ;Get first address
350           JSR SETUP         ;Check it out
351           BCC NOTALTM       ;Not in alt ZP
352           STA SETALTZP
353 NOTALTM   PHA
354           LDX #SWAPPER-MOVER  ;Swap MOVER into ZP
355           LDA #MOVER/
356           LDY #MOVER
357           JSR P@SWAP
358           PLA
359           ASL               ;Set carry if aux RAM
```

LISTING 1: DUBLSTUF Source Code *(continued)*

```
360             TAY
361             LDA #2              ;Finish altering MOVER
362             ADC #0              ;Add val of carry
363             STA $04             ;Fix switch
364             TYA
365             BIT RIWBNK1         ;Assume bank 1
366             CMP #$C0            ;RAM $D000 area?
367             BEQ BANKMOVE
368             STA $08
369             ORA #$1F
370             JSR MFIX2
371             BCS CLEANUP         ;Always
372 BANKMOVE    JSR MFIX            ;Use bank 1 for 1st 4K
373             BIT RIWBNK2         ;Use bank 2 for 2nd 4K
374             JSR MFIX
375             BCS CLEANUP         ;Always
376 SWAP        ASL                 ;See comments for MOVE0
377             STA LO1+2
378             STA SAV2+2
379             LDA #2
380             ADC #0
381             STA SW0+1
382             ADC #2
383             STA SW1+1
384             TYA
385             JSR SETUP
386             BCC NOTALT
387             STA SETALTZP
388 NOTALT      PHA
389             LDX #GETLOC-SWAPPER
390             LDA #SWAPPER/
391             LDY #SWAPPER
392             JSR P0SWAP
393             PLA
394             ASL
395             TAY
396             LDA #2
397             ADC #0
398             STA $1
399             ADC #2
400             STA $13
401             TYA
402             BIT RIWBNK1
403             CMP #$C0
404             BNE NOTALTS
405             JSR SFIX
406             BIT RIWBNK2
407             JSR SFIX
408             BCS CLEANUP
409 NOTALTS     STA $05
410             STA $18
411             ORA #$1F
412             JSR SFIX2
413 CLEANUP     JSR P0UNSWAP        ;Restore normal ZP
414             STA SETSTDZP
415 FIX16K      BIT ROMRIW2         ;Turn ROMs back on
416             CLI                 ;Turn interrupts back on
417 FIX80S      LDA S80STATE        ;Check 80STORE old state
418             BPL RTS3            ;Originally off, so leave off
419             STA S80STORE        ;Turn it on
420 RTS3        RTS
421
422 C           LDX #0              ;'Copy' (use MOVE)
423 A           STX MODE            ;'Auxmove' (use MOVEAUX)
424             STA C80STORE        ;So bank switches will work
425             LDX #0              ;Index for 16K bank switches
426             STX LOC
427             JSR GETLOC          ;Checker subroutine
428             STY A1
429             STA A1+1
430             JSR CHKCOM          ;Get 2nd address (From end)
431             JSR GETLOC
432             STY A2
433             STA A2+1
434             LDA #TO
435             JSR SYNCHR          ;Make sure next char is 'TO'
436             JSR GETLOC
437             STY A4
438             STA A4+1
439             LDX LOC
440             LDY MODE            ;C or A?
441             BNE AUX
442             LDA ROMRIW2,X       ;Must be able to read MOVE
443             JSR MOVE
444             BCS FIX16K          ;Always
445 AUX         JSR CHKCOM          ;Get another char
446             CMP #'A             ;Move to aux mem?
447             BEQ AUXMOVE         ;Carry is set
448             LDA #'M
449             JSR SYNCHR          ;Must be an 'M
450             CLC                 ;Means aux to main
451 AUXMOVE     SEI                 ;ROMs locked out, so no 'rupts
452             LDA RIWBNK2,X
453             JSR MOVEAUX
454             JMP FIX16K          ;Always
455
456 PLOT0       LDA MODE            ;Width of plotting line
457             BEQ PLOT1           ;1 wide else 4 wide
458             LDA HCOLOR1         ;Save all info to restore plot
459             PHA                 ; loc after moving & plotting
460             TXA                 ; to right 3 times
461             PHA
462             TYA
463             PHA
464             LDA LOC
465             PHA
466             LDA HMASK
467             PHA
468             LDX #3
469 GORIGHT     JSR INCRX           ;Move to right 1 place
470             JSR PLOT1           ;Plot a point
471             DEX
472             BNE GORIGHT         ;Do it 3x
473             PLA                 ;Back to original spot
474             STA HMASK
475             PLA
476             STA LOC
477             PLA
478             TAY
479             STY YSAV
480             PLA
481             TAX
482             PLA
483             STA HCOLOR1
484 PLOT1       LDA HCOLOR1         ;Get the running color byte
485             AND #$7F            ;Hi bit off, ACC is +
486 XPLOT       BIT LOC             ;If main mem, turn on - flag
487 CARRIER     JSR ZEROPG          ;our routine on p.0
488             STA WMAINRAM        ;Ensure 'normal' R/W status
489             RTS                 ;Point(s) plotted, boss.
490
491 MFIX        LDA #$D0            ;Fix move from $D000 banks
492             STA $08
493             ORA #$0F            ;Move 4k at a time
494 MFIX2       STA $12
495             BNE CARRIER         ;Always
496
497 SFIX        LDA #$D0            ;Fix swap from $D000 banks
498             STA $05
499             STA $18
500             ORA #$0F
501 SFIX2       STA $1F
502             BNE CARRIER         ;Always
503
504 WAIT        LDA IDBYT1          ;Check for //c
505             BEQ RTS5
506 BLANKING    LDA RDVBLBAR        ;Blanking now?
507             BPL BLANKING        ;Yes, so wait
508 LIVE        LDA RDVBLBAR        ;In live video?
509             BMI LIVE            ;Yes so wait for end
510 RTS5        RTS                 ;Now within 7 CPU cycles of
511                                 ; start of blanking
512 V           LDX #3              ;View p. 1 or p. 2
513             JSR XCHEQUE1
514             CMP #2
515             BEQ VIEW2
516             JSR WAIT
517             STA MIXCLR
518             BPL VIEW1
519
520 HGR         LDA #$20            ;Clear & display DHIRES1
521             JSR ZEROX
522 T           JSR WAIT
523             STA MIXSET
524 VIEW1       STA TXTPAGE1
525             STA S80STORE        ;Leave on for p.1
526             BPL HITIEM
527
528 HGR2        LDA #$40            ;Clear & display DHIRES2
529             JSR ZEROX           ;Zero the page
530 VIEW2       JSR WAIT            ;Wait for no raster
531             STA MIXCLR          ;Full graphics
532             STA TXTPAGE2
533             STA C80STORE        ;So p.2 switch will work
534 HITIEM      STA STIOUDIS        ;So DHIRES will work
535             STA STDHIRES
536             STA SET80VID
537             STA HIRES
538             STA TXTCLR          ;Graphics mode
539             RTS
540
541 ZEROX       STA HPAG            ;Zero a DHIRES page
542             SEC
543 Z           LDX #0              ;Carry clr on entry from parser
544             STX COLOR           ;Black
545             BCS CLEARX
546 CLEAR       JSR GR              ;Skipped if coming from HGR
547 CLEARX      JSR INIT2           ;Setup SHAPE = start mem loc
548             LDA COLOR           ;Cover screen with this color
549             PHA
550 LOOP2       PLA
551 LOOP3       LDX #1
552 SETMEM      STA WMAINRAM,X      ;Alt mem first then main
553             PHA
554             AND #$7F            ;Hi bit off
555             STA (SHAPE),Y       ;Color to screen loc
556             PLA
557             ASL
558             ADC #0              ;Rotate the color mask
559             DEX
560             BPL SETMEM          ;Go back for main mem byte
561             INY                 ;Get next mem byte in alt mem
562             BNE LOOP3
563             PHA
564             INC SHAPE+1
565             LDA SHAPE+1         ;Test for end of screen
566             AND #$1F
567             BNE LOOP2
568             PLA                 ;Get original color back
569             EOR #$FF            ;Produce opposite color for
570             STA COLOR           ; subsequent plots
571             JMP FIX80S
572
573 GR          JSR XCHEQUE         ;Change graphics writing page
574             ASL
575             ASL
576             ASL
577             ASL
578             ASL                 ;1 -> $20 etc.
579             STA HPAG
580             RTS
581
582 * Init GBAS, Y, YSAV, HCOLOR1, MODE, LOC for PLOT, DRAW
583 HPOSN       STA $E2             ;Stolen from A'soft to set
584             STX $E0             ; vertical axis location
585             STY $E1
586             PHA
```

LISTING 1: DUBLSTUF Source Code

```
587             AND #$C0
588             STA GBAS
589             LSR
590             LSR
591             ORA GBAS
592             STA GBAS
593             PLA
594             STA GBAS+1
595             ASL
596             ASL
597             ASL
598             ROL GBAS+1
599             ASL
600             ROL GBAS+1
601             ASL
602             ROR GBAS
603             LDA GBAS+1
604             AND #$1F
605             ORA HPAG
606             STA GBAS+1
607             TXA                 ;Horiz. coordinate lo byte
608             LDX $E1
609             LDY #0
610             STY LOC             ;Assume alt. mem
611 LOOP1       CMP #14             ;Divide coord by 14
612             BCS .+6             ;Product to Y, remainder is
613             CPX #1              ; index for bitmask
614             BCC .+10
615             SBC #14
616             BCS .+3
617             DEX                 ;X = Horiz. coord. hi byte
618             INY
619             BNE LOOP1           ;Always
620             STY YSAV            ;Done, now test for alt/main
621             CMP #7              ;If remainder >6, subtract 7
622             BCC .+6             ; and make LOC = #$FF (main)
623             SBC #7
624             DEC LOC
625             TAX
626             LDA STASH,X         ;Get bitmask
627             AND #$7F            ;Turn off hi bit
628             STA HMASK
629             LDX LOC             ;Now make color mask
630             TYA                 ;Test Y if odd
631             LSR                 ;If odd, will set carry
632             BCC NOTODD          ;Rotate mask 1X each move to R
633             DEX
634             DEX
635 NOTODD      LDA COLOR
636             CPX #0              ;Rotate mask until X = 0
637             BEQ ROTATED
638 ROTATE      ASL                 ;Bit 7 -> carry
639             ADC #0              ;Carry -> bit 0
640             INX
641             BNE ROTATE
642 ROTATED     STA HCOLOR1
643             RTS
644
645 . Plotting/draming routine swapped into p.0
646 PLOTTER     BMI COLCHK
647             STA RCARDRAM        ;Loc = 0
648             STA WCARDRAM
649 COLCHK      EOR (GBAS),Y
650             AND HMASK
651             BNE OK
652             INC COLLIDE
653 OK          EOR (GBAS),Y
654             STA (GBAS),Y
655             STA RMAINRAM
656             RTS
657
658 HCK         JSR CHRGET          ;Gobble next byte
659 HFNS        JSR GETINT          ;Ck syntax and plotting range
660             LDX LINNUM          ;Horiz. coord. lo byte
661             TAY                 ;Horiz. coord. hi byte
662             CPY #2              ;>#$22F (#559) is illegal
663             BNE .+4             ;Ill. quant. will set carry
664             CPX #$30
665             JMP $F6CB           ;Into old HFNS routine
666
667 PLOTSWAP   PHA                 ;Setup to swap PLOTTER into p.0
668             TXA
669             PHA
670             TYA
671             PHA
672             LDA #PLOTTER/
673             LDX #HCK-PLOTTER
674             LDY #PLOTTER
675             JSR P0SWAP
676             PLA
677             TAY
678             PLA
679             TAX
680             PLA
681             RTS
682
683 HPLOT       LDX #0              ;Parse HPLOT or PLOT command
684 PLOT       STX MODE
685             LDX #0              ;Zero collision counter
686             STX COLLIDE
687             STA C80STORE        ;80STORE off, status saved
688             CMP #TO             ;? Continued plot
689             BEQ CONTINUE
690             JSR HFNS            ;Check for valid screen vals
691             JSR PLOTSWAP        ;Move plotting routine to p.0
692             JSR HPOSN           ;Set up coordinates
693             JSR PLOT0           ;Plot the point
694 LOOP5       JSR P0UNSWAP        ;Fix page 0
695             JSR CHRGOT          ;Get last token
696             CMP #TO             ;Continued plot?
697             BEQ CONTINUE
698             JMP FIX80S          ;No, so quit
699 CONTINUE    JSR HCK             ;Do it again
```

```
700             JSR PLOTSWAP
701             STY $9D             ;Section stolen from A'soft
702             TAY                 ; to find intermediate points
703             TXA
704             LDX $9D
705             PHA
706             SEC
707             SBC $E0
708             PHA
709             TXA
710             SBC $E1
711             STA $D3
712             BCS .+$C
713             PLA
714             EOR #$FF
715             ADC #1
716             PHA
717             LDA #0
718             SBC $D3
719             STA $D1
720             STA $D5
721             PLA
722             STA $D0
723             STA $D4
724             PLA
725             STA $E0
726             STX $E1
727             TYA
728             CLC
729             SBC $E2
730             BCC .+6
731             EOR #$FF
732             ADC #$FE
733             STA $D2
734             STY $E2
735             ROR $D3
736             SEC
737             SBC $D0
738             TAX
739             LDA #$FF
740             SBC $D1
741             STA $1D
742             LDY YSAV
743             BCS .+7
744 LOOP9       ASL
745             JSR INTX            ;Move left or right
746             SEC
747             LDA $D4
748             ADC $D2
749             STA $D4
750             LDA $D5
751             SBC #0
752 LOOP6       STA $D5
753             JSR PLOT0           ;Plot the point
754             INX
755             BNE .+6
756             INC $1D
757             BEQ LOOP5
758             LDA $D3
759             BCS LOOP9
760             JSR INTY            ;Move up or down
761             CLC
762             LDA $D4
763             ADC $D0
764             STA $D4
765             LDA $D5
766             ADC $D1
767             BVC LOOP6           ;Always
768
769 XDRAWER     BMI .+8             ;Moved to p. 0 for xdrawing
770             STA RCARDRAM        ;Skipped if minus
771             STA WCARDRAM
772             AND (GBAS),Y
773             BNE .+6             ;No collision
774             INC COLLIDE         ;It was already that way
775             LDA HMASK           ;Don't bother with color
776             EOR (GBAS),Y        ;Flip the bit
777             STA (GBAS),Y        ;Put it back
778             STA RMAINRAM        ;To get home to CARRIER
779             RTS
780
781 DRAWSUB     LDA $D1             ;Does plot and move for DRAW
782             AND #4
783             BEQ NOPLOT
784             LDA MODE
785             BEQ DRAWIT
786             LDA HMASK           ;XDRAW routine
787             JSR XPLOT
788             JMP NOPLOT          ;Skip DRAW routine
789 DRAWIT      JSR PLOT1
790 NOPLOT      LDA $D1
791             ADC $D3
792             AND #3
793             CMP #2
794             ROR
795             BCS INTX
796             JMP INTY            ;Incr. or decr. vert.
797 INTX        BPL INCRX           ;Incr. or decr. horiz. PDQ
798 DECRX       LSR HMASK           ;Next bit over to decrement
799             BCS FIXL            ;Oops! Shifted into carry
800             RTS                 ;It's OK, so get out fast
801 FIXL        LDA LOC             ;Alt <-> main
802             EOR #$FF            ;Flip them bits
803             STA LOC
804             BPL +9
805             DEY                 ;Move 1 mem loc to L
806             BPL +4
807             LDY #$27            ;Wraparound to R side of screen
808             STY YSAV
```

LISTING 1: DUBLSTUF Source Code *(continued)*

```
809           LDA HCOLOR1      ;Rotate the color mask
810           LSR
811           BCC .+4          ;Lo bit was 0
812           ORA #$80         ;Turn on hi bit
813           STA HCOLOR1
814           LDA #$40         ;New bit mask
815           STA HMASK
816           RTS
817 INCRX      ASL HMASK        ;Move bit over 1
818           BMI FIXR         ;If in hi bit
819           RTS
820 FIXR      LDA LOC
821           EOR #$FF
822           STA LOC
823           BMI  +11
824           INY
825           CPY #$28         ;Off end of screen?
826           BCC .+4
827           LDY #0           ;Yes, so wrap to L side
828           STY YSAV
829           LDA HCOLOR1      ;Fix color mask
830           ASL
831           ADC #0           ;Transfer carry to bit 0
832           STA HCOLOR1
833           LDA #1           ;New bit mask
834           STA HMASK
835           RTS
836
837 DRAW      LDX #0
838 XDRAW     STX MODE
839           JSR GETBYT       ;Get the # of shape to be drawn
840           LDA SHAPEV
841           STA SHAPE
842           LDA SHAPEV+1
843           STA SHAPE+1      ;Working area for shape mem loc
844           TXA
845           BEQ ERRR         ;Shape 0 not allowed
846           LDX #0           ;Dummy index
847           CMP (SHAPE,X)    ;Check # shapes in table
848           BCC .+7          ;OK if less or same
849           BEQ .+5
850 ERRR      JMP ERR          ;No such shape #
851           ASL              ;This section taken from A'soft
852           BCC .+5
853           INC SHAPE+1
854           CLC
855           TAY
856           LDA (SHAPE),Y
857           ADC SHAPE
858           TAX
859           INY
860           LDA (SHAPE),Y
861           ADC SHAPEV+1
862           STA SHAPE+1
863           STX SHAPE
864           JSR CHRGOT
865           CMP #$C5         ;"AT" token
866           BNE .+8
867           JSR HCK          ;Parse loc to draw
868           JSR HPOSN        ;Set up for drawing
869           LDA MODE         ;DRAW or XDRAW?
870           BEQ GOPLTSWP     ;DRAW
871           LDA #XDRAWER/    ;Move XDRAWer to p.0
872           LDX #DRAWSUB-XDRAWER
873           LDY #XDRAWER
874           JSR P0SWAP
875           BEQ GETROT       ;Always
876 GOPLTSWP   JSR PLOTSWAP     ;Use PLOTTER for drawing
877 GETROT    LDA ROT
878           TAX              ;Ft. lines stolen from A'soft
879           LSR
880           LSR
881           LSR
882           LSR
883           STA $D3
884           TXA
885           AND #$0F
886           TAX
887           LDY $F5BA,X
888           STY $D0
889           EOR #$0F
890           TAX
891           LDY $F5BB,X
892           INY
893           STY $D2
894           LDY YSAV
895           LDX #0
896           STX COLLIDE
897           STA C80STORE
898           JSR WAIT         ;Do drawing during video blank
899           LDA (SHAPE,X)    ;Get 1st byte of shape
900 LOOPC     STA $D1          ;Borrowed from A'soft
901           LDX #$80
902           STX $D4
903           STX $D5
904           LDX SCALE
905 LOOPD     LDA $D4
906           SEC
907           ADC $D0
908           STA $D4
909           BCC .+7
910           CLC
911           JSR DRAWSUB      ;Plot lo point in byte?
912           CLC
913           LDA $D5
914           ADC $D2
915           STA $D5
916           BCC .+5
917           JSR DRAWSUB      ;Plot hi point in byte?
918           DEX
919           BNE LOOPD
920           LDA $D1
921           LSR
922           LSR
923           LSR
924           BNE LOOPC
925           INC SHAPE        ;Get next byte
926           BNE .+4
927           INC SHAPE+1
928           LDA (SHAPE,X)    ;(X = 0)
929           BNE LOOPC        ;If not 0, keep plotting
930           JMP CLEANUP      ;All done, so fix p0 etc & quit
931
932 INVERSER  STA RMAINRAM,X   ;P 0 sub for INVERSE
933           STA WMAINRAM,X   ;Alt ram first, then main
934 LOOPA     LDA (SHAPE),Y    ;Get a screen byte
935           EOR #$7F         ;Invert low 7 bits
936           STA (SHAPE),Y    ;And put it back
937           INY
938           BNE LOOPA        ;Do 256 bytes
939           DEX
940           BPL INVERSER     ;Go back for main mem
941           RTS
942
943 INVERSE   JSR GR           ;'Negative' a Hi-Res screen
944           LDA #INVERSER/   ;INVERSER to p.0
945           LDX #INVERSE-INVERSER
946           LDY #INVERSER
947           JSR P0SWAP
948           JSR INIT1        ;Start loc to SHAPE
949 LOOPB     LDX #1           ;Do alt mem first
950           JSR ZEROPG       ;Do 512 bytes
951           INC SHAPE+1      ;Get next batch
952           LDA SHAPE+1      ;Ck for all done
953           AND #$1F         ;If results 0, 16K done
954           BNE LOOPB        ;No, not done yet
955           LDA $30          ;Invert the drawing color
956           EOR #$FF
957           STA $30
958           JMP CLEANUP      ;Tidy up when you are done
959
960 W         LDX #192         ;'Widen' a HIRES pic
961           STX SBOSTATE     ;Use as vertical counter
962           LDA #$20
963           STA GBAS+1       ;HIRES p.1 starting address
964           LDY #0
965           STY GBAS
966 LOOPL     LDY #0           ;Beg. each horiz. line
967 LOOP7     LDA #1
968           STA LOC          ;Read mask: 1 bit on
969           LDA (GBAS),Y     ;Get a byte in main mem
970           STA MODE         ;Stash it
971           TAX              ;And save in X
972           LDA #0
973           STA (GBAS),Y     ;Erase main mem
974           TXA
975           STA TXTPAGE2     ;Get aux mem
976           BMI SHIFTED      ;Shifted byte (colors 4-7)
977           JSR WSUB         ;Not shifted
978           STA TXTPAGE1     ;Main mem again
979           LSR LOC          ;Fix read mask
980           LDA #1
981           TAX              ;LOOP 1x
982           JSR WSUB2
983           JSR WSUB1
984           BEQ WEND         ;Always
985 SHIFTED   JSR WSUB1
986           STA TXTPAGE1
987           JSR WSUB
988           LDA MODE         ;Fix for bit 6-> nxt byte bit 0
989           AND #$40         ;Is bit 6 on?
990           BEQ WEND         ;No so skip
991           CPY #$27         ;To right-hand end of line?
992           BCS NXTLIN       ;Don't wrap around
993           INY              ;Get next byte to right
994           STA TXTPAGE2     ; in alt mem
995           LDA #1
996           STA (GBAS),Y     ;To turn on bit 0
997           STA TXTPAGE1
998           DEY              ;Get old byte loc back
999 WEND      INY              ;Get next byte to fix
1000          CPY #$28
1001          BCC LOOP7        ;Not end of line
1002 NXTLIN   JSR INCRY        ;Next line down
1003          DEC SBOSTATE
1004          BNE LOOPL
1005          RTS
1006
1007 WSUB     LDA #3
1008          LDX #4           ;2 bits on, loop 4 times
1009          BNE WSUB2        ;Always
1010 WSUB1    LDA #6
1011          LDX #3
1012 WSUB2    STA HMASK        ;Write 2 bits at a time
1013 LOOPB    LDA MODE
1014          AND LOC          ;Is desired bit on?
1015          BEQ SHIFT        ;No, so don't write
1016          LDA HMASK        ;Get writebyte
1017          ORA (GBAS),Y     ;Add to contents of byte
1018          STA (GBAS),Y     ; and save result
1019 SHIFT    ASL LOC          ;Next bit to read
1020          ASL HMASK
1021          ASL HMASK        ;Next 2 bits to write
1022          DEX              ;Result: 1 bit on -> 2 bits on
1023          BNE LOOPB        ;Do rest of bits
1024          RTS              ;All bits accounted for, sir
1025
1026 OUT0     STA LINNUM+1     ;Send built-in commands
1027 OUT1     STX LINNUM       ; to printer
1028 OUT2     LDY #0
1029 LOOPE    LDA (LINNUM),Y   ;Get a byte to send
```

LISTING 1: DUBLSTUF Source Code

```
1030            PHP             ;Save +/- status
1031            ORA #$80        ;Turn on hi bit
1032            JSR COUT
1033            INC LINNUM      ;Get next byte
1034            PLP
1035            BPL LOOPE       ;If hi bit was off
1036            RTS             ;Hi bit on, so quit
1037
1038 PRINTER    BEQ .+5         ;Moved to alt p. 0
1039            STA RCARDRAM    ;Skip if main mem wanted
1040            LDA ($F),Y      ;Get an indexed byte
1041            STA RMAINRAM    ;To get back home
1042            RTS
1043            DFC 0,0,0,0,0,0 ;Space saved for base addresses
1044 PRINT      PHA
1045            LDA #INIT/      ;DHIRES printer dump
1046            LDX #INIT       ;Set printer for 6/72 spacing
1047            JSR OUT0
1048            LDA PINS        ;Make flip value
1049            ORA PINS+1
1050            ORA PINS+2
1051            STA FLIP+1
1052            PLA
1053            STA C80STORE    ;So switches will work
1054 MORE       JSR GR          ;Print which page?
1055            STA SETALTZP    ;Use alt z pg for our stuff
1056            STA HPAG        ;Used by INCRY
1057            STA GBAS+1      ;Set up initial base address
1058            LDA #192/3      ;Loop counter
1059            STA S80STATE
1060            LDA #PRINTER/   ;Move PRINTER to alt p.0
1061            LDX #PRINT-PRINTER
1062            LDY #PRINTER
1063            JSR P8SWAP
1064            STX GBAS
1065 LOOPF      LDX #LINE       ;Apple to printer:
1066            STA SETSTDZP    ; "graphics coming"
1067            JSR OUT1
1068            STA SETALTZP
1069            LDX #6          ;Get 3 base addresses
1070 LOOPG      LDA GBAS+1      ;And save them on p.0
1071            STA $A,X
1072            DEX
1073            LDA GBAS
1074            STA $A,X
1075            JSR INCRY       ;Get next base address
1076            DEX
1077            BNE LOOPG
1078 LOOPH      LDA #1
1079            STA LOC         ;Alt first then main
1080 LOOPI      LDA #1
1081            STA MASK+1      ;Start with bit 0
1082 LOOPJ      LDA #0
1083            STA HMASK       ;Stash for pins to hit
1084            LDX #3          ;Point PRINTER to our addresses
1085            LDA #$F
1086            STA $6
1087 LOOPK      LDA LOC         ;Alt or main
1088            JSR ZEROPG      ;Get a byte
1089 MASK       AND #1          ;Mask all but 1 bit
1090            BEQ .+5         ;If 0, do nothing
1091            LDA PINS-1,X    ;If 1, get pins to hit
1092            ORA HMASK       ;Combine with prior pins
1093            STA HMASK       ;And save
1094            DEC $6          ;Get a different address
1095            DEC $6
1096            DEX             ;Our counter
1097            BNE LOOPK       ;Get 3 bits -> 6 pins
1098            STA SETSTDZP    ;To printer soon
1099            LDX INVFLG      ;Invert picture?
1100            BPL .+4         ;Yes if INVERSE
1101 FLIP       EOR #$3F        ;Make white -> white
1102            LDX PINS        ;Check printer type
1103            CPX #3          ;Epson lower pins
1104            BNE NOTEPSON
1105            TAX
1106            LDA MODE        ;Odd/even counter
1107            LSR             ;Odd sets carry
1108            TXA
1109            BCC EVEN
1110            JSR COUT        ;Hit odd pins 2X in 960 mode
1111 EVEN       DEC MODE
1112 NOTEPSON JSR COUT
1113            STA SETALTZP    ;To get more bits
1114            ASL MASK+1      ;Get next higher bit
1115            BPL LOOPJ       ;Don't use bit 7
1116            DEC LOC         ;Now for main memory
1117            BPL LOOPI       ;Do it all again
1118            LDA $7F8        ;Old Apple II card fix
1119            BNE .+5
1120            STA $7F9        ;Zero column count so no CR
1121            INY             ;Next byte pair
1122            CPY #$28        ;To end of line yet?
1123            BCC LOOPH
1124            DEC S80STATE    ;Our counter
1125            BEQ KAY         ;All done
1126            JMP LOOPF       ;Too far to branch back
1127 KAY        JSR P8UNSWAP    ;Rescue PRINTER
1128            STA SETSTDZP
1129            JSR CHRGOT      ;Check for another page
1130            CMP #$2C        ;A comma?
1131            BNE RESTOR      ;No, so done
1132            JSR CHRGET
1133            JMP MORE        ;Yes, so do it again
1134 RESTOR     JSR OUT2        ;Restore printer normal state
1135            JMP FIX80S      ;Out of memory, so stop!
1136
1137 PINS       DFC $30,$C,3    ;DMP, Imagewriter, etc.
1138 INIT       DFC $1B,$3E,$1B,$6E,$1B,$54,$31,$82
1139 LINE       DFC $D,$20,$18,$47,$30,$35,$36,$80
1140            DFC $1B,$3C,$1B,$4E,$1B,$C1
1141 SHTBL      DFC 1,0,4,0,$C,$2D,$3C,$3C,$FC,$6F,$D,$2D ;Star
1142            DFC $2D,$3C,$3F,$C,$25,$15,$15,$2F,$2D,$2D,$DE
1143            DFC $3B,$3F,$3F,$AF,$2D,$2D,$3E,$D,$E,$4D,0,0
```

END OF LISTING 1

```
KEY PERFECT 5.0
RUN ON
DUBLSTUF
```

CODE-5.0	ADDR# - ADDR#	CODE-4.0
C04940A4	318A - 31D9	2677
2DC64917	31DA - 3229	2852
BED400F4	322A - 3279	27D3
42C8C949	327A - 32C9	2622
760F2106	32CA - 3319	27C8
3C41E953	331A - 3369	243D
6789EDBA	336A - 33B9	27F0
DCD1E3A8	33BA - 3409	297D
831A63F0	340A - 3459	28C9
43B4F852	345A - 34A9	29D2
8D3A27B6	34AA - 34F9	2759
F3D51DF2	34FA - 3549	25A3
3007A658	354A - 3599	2C24
D4AE3FFA	359A - 35E9	2B11
AEEA9652	35EA - 3639	2A35
FAAFE51D	363A - 3689	2B58
20773BDC	368A - 36D9	2912
BBB9EFD2	36DA - 3729	26C8
695938BF	372A - 3779	225C
150A9D36	377A - 37C9	2525
7F4668F6	37CA - 3819	26A7
3E36CB6F	381A - 3869	2C7D
1A69D74B	386A - 38B9	25D2
CDFBC190	38BA - 3909	29D6
8A67768B	390A - 3959	268A
DB25382F	395A - 39A9	2702
A2EEE2E8	39AA - 39F8	278F
A25FAB06	= PROGRAM TOTAL =	086F

LISTING 2: DUBLSTUF Object Code

```
318A- AD FF BF 30 14 AD
3190- 98 BF 29 10 F0 0A 20 F8
3198- BE A9 08 20 F5 BE 90 1A
31A0- 4C 10 D4 AD B3 FB C9 06
31A8- D0 F6 A9 01 20 58 A2 A9
31B0- 00 A0 92 85 50 84 51 20
31B8- 8E F2 A9 20 85 E6 A0 FF
31C0- 84 30 84 3E C8 84 3C 84
31C8- 42 84 F9 A9 32 85 3D A9
31D0- 39 85 3F A9 92 85 43 20
31D8- 2C FE A9 D6 85 E8 A9 99
31E0- 85 E9 C8 84 E7 AD F7 03
31E8- C9 92 F0 13 8D 0F 92 AD
31F0- F6 03 8D 0E 92 A9 00 8D
31F8- F6 03 A9 92 8D F7 03 60
3200- 2C 81 C0 A2 15 DD 36 92
3208- F0 06 CA D0 F8 4C 58 FF
3210- 08 A9 92 48 A9 2F 48 8A
3218- 0A AA BD 4B 92 48 CA BD
3220- 4B 92 48 AD 18 C0 85 FE
3228- 10 03 8D 54 C0 4C B1 00
3230- 28 20 B7 00 4C 95 D9 41
3238- 43 46 4D 50 53 54 56 57
3240- 5A 88 8D 90 91 93 94 95
3248- 9E B5 BA BD 5E 94 5C 94
3250- 7B 92 8B 93 80 92 8D 93
3258- 1D 95 07 95 46 98 4F 95
3260- 84 95 36 96 2B 95 18 95
3268- 34 96 5A 97 5C 97 21 98
3270- F7 94 ED 98 55 95 20 67
3278- DD 4C 52 E7 8D 52 C0 58
3280- 60 20 76 92 20 BE DE 20
3288- 67 DD 20 08 E1 E6 A4 A5
3290- A1 D0 02 C6 A0 AD C0 FB
3298- 78 A6 A1 A4 A0 30 0A 2C
32A0- 30 C0 C9 00 F0 03 2C 20
32A8- C0 C6 A4 D0 08 C6 50 D0
32B0- 04 C6 51 30 CA CA D0 F1
32B8- 88 30 DE 10 EC CA 8E CE
32C0- 92 8C D0 92 8C D8 92 8D
32C8- D1 92 8D D9 92 A2 00 BD
32D0- FF FF B4 00 95 00 98 9D
32D8- FF FF CA 10 F2 E8 60 A5
32E0- E6 85 1B A0 00 84 1A 8D
32E8- 00 C0 60 8D 04 C0 8D 02
32F0- C0 BD 00 60 9D 00 40 E8
32F8- D0 F7 A5 08 C9 5F E6 08
3300- E6 0B 90 ED 8D 02 C0 60
3308- 8D 02 C0 BD 00 60 8D 02
3310- C0 BC 00 40 8D 04 C0 9D
3318- 00 40 8D 04 C0 98 9D 00
3320- 60 E8 D0 E4 A5 05 C9 5F
3328- E6 05 E6 0B E6 11 E6 18
3330- 90 D6 8D 02 C0 60 20 76
3338- 92 C9 C0 90 0F AE FF BF
3340- 10 36 C9 D0 B0 06 A2 08
3348- 86 FC 09 10 60 A2 04 86
3350- 9D 20 F8 E6 8A F0 21 C5
3358- 9D B0 1D 60 0A 0A 0A 0A
3360- C9 40 90 13 C9 60 90 10
3368- C9 80 F0 0C C9 90 AE FF
3370- BF 30 02 90 03 C9 E0 60
3378- 8D 08 C0 AD 1F C0 10 03
3380- 8D 01 C0 8D 54 C0 8D 51
3388- C0 4C 99 E1 A2 00 86 FB
3390- A2 10 20 4F 93 85 EA A9
3398- C1 20 C0 DE 20 51 93 C9
33A0- 06 F0 D5 C9 0E F0 D1 8D
33A8- 00 C0 20 5C 93 A4 EA A6
33B0- FB 78 90 03 8D 09 C0 D0
33B8- 43 0A 8D F6 92 A9 04 69
33C0- 00 8D EC 92 98 20 5C 93
33C8- 90 03 8D 09 C0 48 A2 1D
33D0- A9 92 A0 EB 20 BD 92 68
33D8- 0A A8 A9 02 69 00 85 04
33E0- 98 2C 8B C0 C9 C0 F0 09
33E8- 85 08 09 1F 20 E8 94 B0
33F0- 5A 20 E2 94 2C 83 C0 20
33F8- E2 94 B0 4F 0A 8D 13 93
3400- 8D 19 93 A9 02 69 00 8D
3408- 0F 93 69 02 8D 15 93 98
3410- 20 5C 93 90 03 8D 09 C0
3418- 48 A2 2E A9 93 A0 08 20
3420- BD 92 68 0A A8 A9 02 69
3428- 00 85 01 69 02 85 13 98
3430- 2C 8B C0 C9 C0 D0 0B 20
3438- EC 94 2C 83 C0 20 EC 94
3440- B0 09 85 05 85 18 09 1F
3448- 20 F4 94 20 CD 92 8D 08
3450- C0 2C 81 C0 58 A5 FE 10
3458- 03 8D 01 C0 60 A2 00 86
3460- FB 8D 00 C0 A2 00 86 FC
3468- 20 36 93 84 3C 85 3D 20
3470- BE DE 20 36 93 84 3E 85
3478- 3F A9 C1 20 C0 DE 20 36
3480- 93 84 42 85 43 A6 FC A4
3488- FB D0 08 BD 81 C0 20 2C
3490- FE B0 BE 20 BE DE C9 41
3498- F0 06 A9 4D 20 C0 DE 18
34A0- 78 BD 83 C0 20 11 C3 4C
34A8- 51 94 A5 FB F0 27 A5 1C
34B0- 48 8A 48 98 48 A5 FC 48
34B8- A5 FD 48 A2 03 20 39 97
34C0- 20 D5 94 CA D0 F7 68 85
34C8- FD 68 85 FC 68 A8 84 E5
34D0- 68 AA 68 85 1C A5 1C 29
34D8- 7F 24 FC 20 00 00 8D 04
34E0- C0 60 A9 D0 85 08 09 0F
34E8- 85 12 D0 EF A9 D0 85 05
34F0- 85 18 09 0F 85 1F D0 E3
34F8- AD C0 FB F0 0A AD 19 C0
3500- 10 FB AD 19 C0 30 FB 60
3508- A2 03 20 4F 93 C9 02 F0
3510- 20 20 F8 94 8D 52 C0 10
3518- 0B A9 20 20 4D 95 20 F8
3520- 94 8D 53 C0 8D 54 C0 8D
3528- 01 C0 10 11 A9 40 20 4D
3530- 95 20 F8 94 8D 52 C0 8D
3538- 55 C0 8D 00 C0 8D 7E C0
3540- 8D 5E C0 8D 0D C0 8D 57
3548- C0 8D 50 C0 60 85 E6 38
3550- A2 00 86 30 B0 03 20 85
3558- 95 20 E1 92 A5 30 48 68
3560- A2 01 9D 04 C0 48 29 7F
3568- 91 1A 68 0A 69 00 CA 10
3570- F1 C8 D0 EC 48 E6 1B A5
3578- 1B 29 1F D0 E2 68 49 FF
3580- 85 30 4C 55 94 20 4D 93
3588- 0A 0A 0A 0A 0A 85 E6 60
3590- 85 E2 86 E0 84 E1 48 29
3598- C0 85 26 4A 4A 05 26 85
35A0- 26 68 85 27 0A 0A 0A 26
35A8- 27 0A 26 27 0A 66 26 A5
35B0- 27 29 1F 05 E6 85 27 8A
35B8- A6 E1 A0 00 84 FC C9 0E
35C0- B0 04 E0 01 90 08 E9 0E
35C8- B0 01 CA C8 D0 F0 84 E5
```

```
35D0- C9 07 90 04 E9 07 C6 FC
35D8- AA BD B2 F5 29 7F 85 FD
35E0- A6 FC 98 4A 90 02 CA CA
35E8- A5 30 E0 00 F0 06 0A 69
35F0- 00 E8 D0 FA 85 1C 60 30
35F8- 06 8D 03 C0 8D 05 C0 51
3600- 26 25 FD D0 02 E6 EA 51
3608- 26 91 26 8D 02 C0 60 20
3610- B1 00 20 76 92 A6 50 A8
3618- C0 02 D0 02 E0 30 4C CB
3620- F6 48 8A 48 98 48 A9 95
3628- A2 18 A0 F7 20 BD 92 68
3630- A8 68 AA 68 60 A2 00 86
3638- FB A2 00 86 EA 8D 00 C0
3640- C9 C1 F0 19 20 12 96 20
3648- 21 96 20 90 95 20 AA 94
3650- 20 CD 92 20 B7 00 C9 C1
3658- F0 03 4C 55 94 20 0F 96
3660- 20 21 96 84 9D A8 8A A6
3668- 9D 48 38 E5 E0 48 8A E5
3670- E1 85 D3 B0 0A 68 49 FF
3678- 69 01 48 A9 00 E5 D3 85
3680- D1 85 D5 68 85 D0 85 D4
3688- 68 85 E0 86 E1 98 18 E5
3690- E2 90 04 49 FF 69 FE 85
3698- D2 84 E2 66 D3 38 E5 D0
36A0- AA A9 FF E5 D1 85 1D A4
36A8- E5 B0 05 0A 20 15 97 38
36B0- A5 D4 65 D2 85 D4 A5 D5
36B8- E9 00 85 D5 20 AA 94 E8
36C0- D0 04 E6 1D F0 8A A5 D3
36C8- B0 E1 20 D3 F4 18 A5 D4
36D0- 65 D0 85 D4 A5 D5 65 D1
36D8- 50 E0 30 06 8D 03 C0 8D
36E0- 05 C0 31 26 D0 04 E6 EA
36E8- A5 FD 51 26 91 26 8D 02
36F0- C0 60 A5 D1 29 04 F0 0F
36F8- A5 FB F0 08 A5 FD 20 D9
3700- 94 4C 07 97 20 D5 94 A5
3708- D1 65 D3 29 03 C9 02 6A
3710- B0 03 4C D3 F4 10 22 46
3718- FD B0 01 60 A5 FC 49 FF
3720- 85 FC 10 07 88 10 02 A0
3728- 27 84 E5 A5 1C 4A 90 02
3730- 09 80 85 1C A9 40 85 FD
3738- 60 06 FD 30 01 60 A5 FC
3740- 49 FF 85 FC 30 09 C8 C0
3748- 28 90 02 A0 00 84 E5 A5
3750- 1C 0A 69 00 85 1C A9 01
3758- 85 FD 60 A2 00 86 FB 20
3760- F8 E6 A5 E8 85 1A A5 E9
3768- 85 1B 8A F0 08 A2 00 C1
3770- 1A 90 05 F0 03 4C 78 93
3778- 0A 90 03 E6 1B 18 A8 B1
3780- 1A 65 1A AA C8 B1 1A 65
3788- E9 85 1B 86 1A 20 B7 00
3790- C9 C5 D0 06 20 0F 96 20
3798- 90 95 A5 FB F0 0B A9 96
37A0- A2 18 A0 DA 20 BD 92 F0
37A8- 03 20 21 96 A5 F9 AA 4A
37B0- 4A 4A 4A 85 D3 8A 29 0F
37B8- AA BC BA F5 84 D0 49 0F
37C0- AA BC BB F5 C8 84 D2 A4
37C8- E5 A2 00 86 EA 8D 00 C0
37D0- 20 F8 94 A1 1A 85 D1 A2
37D8- 80 86 D4 86 D5 A6 E7 A5
37E0- D4 38 65 D0 85 D4 90 05
37E8- 18 20 F2 96 18 A5 D5 65
37F0- D2 85 D5 90 03 20 F2 96
37F8- CA D0 E4 A5 D1 4A 4A 4A
3800- D0 D3 E6 1A D0 02 E6 1B
3808- A1 1A D0 C9 4C 4B 94 9D
3810- 02 C0 9D 04 C0 B1 1A 49
3818- 7F 91 1A C8 D0 F7 CA 10
3820- EE 60 20 85 95 A9 98 A2
3828- 13 A0 0F 20 BD 92 20 DF
3830- 92 A2 01 20 00 00 E6 1B
3838- A5 1B 29 1F D0 F3 A5 30
3840- 49 FF 85 30 4C 4B 94 A2
3848- C0 86 FE A9 20 85 27 A0
3850- 00 84 26 A0 00 A9 01 85
3858- FC B1 26 85 FB AA A9 00
3860- 91 26 8A 8D 55 C0 30 13
3868- 20 A7 98 8D 54 C0 46 FC
3870- A9 01 AA 20 B1 98 20 AD
```

LISTING 2: DUBLSTUF Object Code

(continued)

```
3878- 98 F0 1F 20 AD 98 8D 54
3880- C0 20 A7 98 A5 FB 29 40
3888- F0 10 C0 27 B0 11 C8 8D
3890- 55 C0 A9 01 91 26 8D 54
3898- C0 88 C8 C0 28 90 B6 20
38A0- 04 F5 C6 FE D0 AD 60 A9
38A8- 03 A2 04 D0 04 A9 06 A2
38B0- 03 85 FD A5 FB 25 FC F0
38B8- 06 A5 FD 11 26 91 26 06
38C0- FC 06 FD 06 FD CA D0 EB
38C8- 60 85 51 86 50 A0 00 B1
38D0- 50 08 09 80 20 ED FD E6
38D8- 50 28 10 F3 60 F0 03 8D
38E0- 03 C0 B1 0F 8D 02 C0 60
38E8- 00 00 00 00 00 00 48 A9
38F0- 99 A2 C0 20 C9 98 AD BD
38F8- 99 0D BE 99 0D BF 99 8D
3900- 6D 99 68 8D 00 C0 20 85
3908- 95 8D 09 C0 85 E6 85 27
3910- A9 40 85 FE A9 98 A2 11
3918- A0 DD 20 BD 92 86 26 A2
3920- C8 8D 08 C0 20 CB 98 8D
3928- 09 C0 A2 06 A5 27 95 0A
3930- CA A5 26 95 0A 20 04 F5
3938- CA D0 F1 A9 01 85 FC A9
3940- 01 8D 54 99 A9 00 85 FD
3948- A2 03 A9 0F 85 06 A5 FC
3950- 20 00 00 29 01 F0 03 BD
3958- BC 99 05 FD 85 FD C6 06
3960- C6 06 CA D0 E9 8D 08 C0
3968- A6 32 10 02 49 3F AE BD
3970- 99 E0 03 D0 0C AA A5 FB
3978- 4A 8A 90 03 20 ED FD C6
3980- FB 20 ED FD 8D 09 C0 0E
3988- 54 99 10 B8 C6 FC 10 AF
3990- AD F8 07 D0 03 8D F9 07
3998- C8 C0 28 90 9E C6 FE F0
39A0- 03 4C 1F 99 20 CD 92 8D
39A8- 08 C0 20 B7 00 C9 2C D0
39B0- 06 20 B1 00 4C 06 99 20
39B8- CD 98 4C 55 94 30 0C 03
39C0- 1B 3E 1B 6E 1B 54 31 B2
39C8- 0D 20 1B 47 30 35 36 B0
39D0- 1B 3C 1B 4E 1B C1 01 00
39D8- 04 00 0C 2D 3C 3C FC 6F
39E0- 0D 2D 2D 3C 3F 0C 25 15
39E8- 15 2E 2D 2D DE 3B 3F 3F
39F0- AF 2D 2D 3E 0D 0E 4D 09
39F8- 00
```

END OF LISTING 2

LISTING 3: DUBL.DEMO

```
10  REM ************************
20  REM *      DUBL.DEMO         *
30  REM *  David L. Smith MD    *
40  REM * Copyright (c) 1987   *
50  REM * by MicroSPARC, Inc.  *
60  REM * Concord, MA  01742   *
70  REM ************************
80  IF  PEEK (64435) <  > 6 THEN  HOME : VTAB
    12: PRINT "SORRY, PROGRAM WON'T WORK ON"
    : PRINT "THIS MACHINE": END : REM MUST B
    E IIC, IIGS OR 128K IIE
90  ONERR  GOTO 350
100  PRINT  CHR$ (4)"BRUN DUBLSTUF": POKE 216
    ,0: GOTO 130
110  HOME : PRINT : HTAB (80 -  LEN (X$)) / 2
    : VTAB 22: PRINT X$: & P500,0: RETURN
120  READ D: READ P: & PD,P: RETURN
130 D$ =  CHR$ (4): PRINT D$"PR#3": PRINT : VTAB
    22
140  HGR :X$ = "This is single-HIRES graphics
    ": GOSUB 110
150 C = 0: FOR Y = 0 TO 130 STEP 26:C = C + 1
    : HCOLOR= C: FOR Z = Y TO Y + 25: HPLOT
    0,Z TO 279,Z: NEXT : NEXT
160 X$ = "Let's double it with &W": GOSUB 110
    : & M1 TO 2: &  HGR : & M2 TO 1: & W: &
    P1000,0
170 X$ = "Now let's check the DHIRES colors":
```

```
     GOSUB 110: & Z1
180  FOR C = 1 TO 15: COLOR= C: IF C = 13 THEN
     & F
190 Y = C + 12: FOR Z = Y TO Y + 11: &  HPLOT
     0,Z TO 559,Z: NEXT : NEXT : FOR T = 0 TO
     32: &  XDRAW 1 AT T + 17,Y + 9:CLICK =  PEEK
     (49200) + PEEK (49200) + PEEK (49200):
      NEXT
200  & M1 TO 2: & M9 TO 15: & S14 TO 2
210  & T:X$ = "Here's a demo of &CLEAR:": GOSUB
     110
220  & Z1: FOR C = 1 TO 15: COLOR= C: &  CLEAR
     1: NEXT
230 X$ = "Now for some patriotism demo'ing &H
     PLOT, &P, &PLOT, &DRAW ": GOSUB 110: COLOR=
     15: &  CLEAR 2: & V2
240  COLOR= 12: FOR Y = 15 TO 135 STEP 20: FOR
     Z = Y TO Y + 9: REM Old glory
250  &  HPLOT 0,Z TO 559,Z: NEXT : NEXT
260  COLOR= 1: FOR X = 0 TO 232 STEP 4: &  PLOT
     X,15 TO X,84: ON X > 210 GOSUB 120: NEXT

270  COLOR= 15: FOR X = 14 TO 209 STEP 39: FOR
     Y = 25 TO 81 STEP 14: &  DRAW 1 AT X,Y: GOSUB
     120: NEXT : NEXT
280  FOR X = 34 TO 190 STEP 39: FOR Y = 31 TO
     73 STEP 14: &  DRAW 1 AT X,Y: GOSUB 120:
      NEXT : NEXT : & M2 TO 1: & M10 TO 9: &
     T
290 X$ = "Now for a disappearing act": GOSUB
     110: & M10 TO 3: &  HGR : & P1000,0
300 X$ = "Let's get our flag back the slow wa
     y...": GOSUB 110: & C16384,24570 TO 8192
     : & A24576,32760 TO 8192,A: & P1000,0: &
      HGR
310 X$ = "Here's the fast way to do it:": GOSUB
     110: & M2 TO 1: & M10 TO 9: & P1000,0
320 X$ = "Here's the &INVERSE command": GOSUB
     110: &  INVERSE 1: & P1000,0: &  INVERSE
     1: & P1000,0
330 X$ = "Here's the &S command:": GOSUB 110:
     & S14 TO 1: & S9 TO 15: & P1000,0: &  WAIT
     : & S14 TO 1: & S9 TO 15
340 X$ = "End of Demonstration": GOSUB 110: END

350 E =  PEEK (222):EL =  PEEK (218) + 256 *
     PEEK (219): CALL  - 3288: POKE 216,0
360  HOME : VTAB 12: IF E = 6 THEN  PRINT "UN
     ABLE TO FIND DUBLSTUF ON THIS DISK": GOTO
     390
370  IF E = 8 THEN  PRINT "I/O ERROR--CHECK D
     RIVE DOOR": GOTO 390
380  PRINT "ERROR "E" IN LINE "EL
390  VTAB 21: PRINT "RETURN TO TRY AGAIN,ESCA
     PE TO QUIT";: GET Z$: PRINT : IF Z$ =  CHR$
     (13) GOTO 90
400  DATA 64,164,96,164,32,195,64,195,64,164,
     96,164,32,219,64,219,64,195,64,184,64,16
     4,64,146,64,130,192,164
410  DATA 64,164,96,164,32,195,64,195,64,164,
     96,164,32,219,64,219,64,110,64,116,64,11
     0,64,98,64,146,192,110
420  DATA 64,164,96,98,32,98,64,110,64,123,96
     ,123,32,130,64,130,64,123,64,110,64,130,
     64,146,64,164,192,123
430  DATA 64,123,96,123,32,146,64,146,64,123,
     96,123,32,164,64,164,64,164,64,146,64,12
     3,64,164,64,110,192,123
```

END OF LISTING 3

```
          KEY PERFECT 5.0
              RUN ON
           DUBL.DEMO
=========================================
   CODE-5.0   LINE# - LINE#   CODE-4.0
   --------   -----------   --------
   4558BAA5     10 -   100      8843
   7EEC573A    110 -   200      C3EA
   924E9730    210 -   300      F089
   9CBAD18E    310 -   400      010F91
   97461C23    410 -   430      964C
   6FE06001 = PROGRAM TOTAL =    0825
```