GS SOFT SWITCHES

Use the IIGS soft switches to gain hardware-level control of your machine.

The IIGS offers a great number of improvements over the IIe. The most highly touted ones are the 320 x 200 and 640 x 200 Super Hi-Res graphics modes and the Ensoniq sound chip, which supports both synthesized and digitized sound. Some of the other improvements have gotten less billing, but are exciting in their own right. For example, a number of soft switches have been added to control the machine and detect its condition.

SOFT SWITCHES

Most readers who have programmed the earlier Apples are familiar with soft switches. For instance, it takes three of them to display Hi-Res page 2 without clearing it. It takes another to clear the keyboard strobe. Soft switches, while occupying address space in the range 49152-49295 ($C000-$C08E), are neither RAM nor ROM, but a special kind of memory called input/output (I/O). This is how the Apple communicates with the outside world — via the keyboard, the mouse, the internal clock, the screen display, the sound chip, the game controls, and storage devices such as disk drives and RAM cards. Many of these locations are manipulated only by built-in Apple programs and should not be disturbed by the programmer, but many other locations can be used to set or detect various machine conditions.

This article barely touches the surface of what can be done by reading and manipulating the soft switches. To explore further, see Apple/Addison-Wesley's *Apple IIGS Firmware Reference Manual* (1987) or Michael Fisher's *Apple IIGS Technical Reference* (Osborne/McGraw-Hill, 1987).

DETECTING MODIFIER KEYS

With the IIGS, it's easy to tell what keys are being held down. As with previous machines, register 49152 (−16384 or $C000) holds the ASCII code for the character being generated by a keypress combination. For instance, if you press the F key in response to a GET, register 49152 holds 102 ($66), the ASCII code for a lower-case f; with the Shift key down, the code becomes 70 ($46). However, when you press the Caps Lock key and then press F, the code is still 70. The IIGS provides an easy way to tell which modifier keys (Shift, Caps Lock, Open Apple, Option, and Control) are being held down. In addition, it can tell the difference between the 1 key on the keypad and the 1 key on the main keyboard.

GS.MODIFIERS demonstrates the technique used to determine which keys are being pressed. To use the program (see **Listing 1**), just RUN GS.MODIFIERS and press key combinations. The program will then display words describing the keys you're holding down. One of the keys you press must be a nonmodifier key, such as a letter, number or symbol. The alphabetic keys are represented by capital letters, since that's what is shown on the keys.

Certain key combinations will cause unexpected results. If you hold down the Control key and press an arrow key, Return, Escape or Tab, the program will print the combination as Control H, U, J, K, M, [, or I. The reason is that these keys with the Control key also generate the special control characters listed; based on the information available, the program can't distinguish between Control-H and Control-Right-Arrow. To stop the program, press Control-Reset, which is the only key combination that won't be displayed.

HOW THE PROGRAM WORKS

Line 90 of **Listing 1** uses a GET command to wait for a keypress. When the keypress has occurred, the variable KEY$ holds the character generated. MOD is used to hold the contents of 49189, the Modifiers Register. Since Applesoft doesn't have bit arithmetic, the values of the relevant bits must be extracted by a test and subtraction routine starting at **line 360**. It returns with flags for each bit in the array BIT( ). See **Table 1** for the functions of the bits. **In lines 110-150**, the flags that represent the modifier key bits are tested and the appropriate key descriptions are printed.

Line 160 tests for the null string, which is generated only when Control-@ is pressed. An ILLEGAL QUANTITY error would be generated if the ASC( ) function **(line 170)** were allowed to operate

TABLE 1: Bits in the Modifiers Register

Bit Number	Function
0	Shift key
1	Control key
2	Caps Lock key
3	Key is being repeated
4	Key is on the keypad
5	Byte is updated without a keypress
6	Option key
7	Apple/Command key

TABLE 2: The 16 Colors Used for Lo-Res, Double Lo-Res, Double Hi-Res, and the IIGS Border, Text and Background

Dec	Hex	Color	Dec	Hex	Color
0	0	Black	8	8	Brown
1	1	Deep red	9	9	Orange
2	2	Dark blue	10	A	Light gray
3	3	Purple	11	B	Pink
4	4	Dark green	12	C	Light green
5	5	Dark gray	13	D	Yellow
6	6	Medium blue	14	E	Aquamarine
7	7	Light blue	15	F	White

on a null string. All other characters are converted to their ASCII values in **line 170**. **Line 180** tests for control characters, which are sent to the subroutine at **line 450**. Several of these characters have special names. Keys that have special names skip the remaining tests and are printed in **line 320**.

Keys that don't have special names return with 64 added into the value of KEY. This is converted to a character in **line 310** before it is printed out. For instance, the subroutine would identify the Left-Arrow as a special key and print "Left Arrow" rather than "Control-H".

Line 190 tests the keypad flag and skips the remaining tests. **Line 200** checks any key with a code in the range 33-43. Most of these are obtained by pressing Shift with another key. The subroutine at **line 580** converts these shift characters to the equivalent unshifted key value. **Lines 210-290** sort out the remaining keys with codes from 44-127. Codes 44-57, 91-94 and the upper-case letters (codes 65-90) can be printed as they are (**line 210**), but most of the others must be dealt with on an individual basis. By the time KEY gets to **line 300**, all that's left are the upper-case letters — and they're converted to their upper-case equivalents by subtracting 32. The final value of KEY is converted to a character in **line 310** and printed in **line 320**.

Two bits in the Modifiers Register have special purposes: bit 4 signals a keypad key (**line 330**) and bit 3 means that the key combination has been repeated by holding the key(s) down (**line 340**). In **line 350**, the program issues a final PRINT command and returns to line 90 for the next key combination.

GS SPEED

The GS has two operation speeds: normal and fast. For most applications, you can use the fast speed, but for some — particularly applications that use animation or sound produced with an older Apple model in mind — the normal speed is required. One way to determine the speed is to go to the Control Panel (Open-Apple-Control-Escape) and check the system speed setting. Another way is to use the technique shown in **Listing 2.** It reads Register 49206 ($C036) and checks if the value is greater than 127. If it is, the machine is running in fast mode.

MONOCHROME HI-RES

Another feature of the IIGS is its capability to display the regular Hi-Res screen in monochrome. This means that you make use of the full resolution of the Hi-Res screen without encountering colored aberrations when dots are plotted too closely together. **Listing 3** demonstrates this by filling the screen with full-width horizontal lines. It then switches into double Hi-Res, but without switching to 80-columns. When you enter the Control Panel and switch to monochrome, the colored lines turn into lines of white dots.

BLINKY

The IIGS, the IIc, and enhanced IIe provide greater support for a previously underutilized capability of the machines' processors: interrupts. Common sources of interrupts are keystrokes, mouse movements and timer pulses.

The program in **Listing 4** uses an interrupt generated by the video processing circuitry called the vertical blank interrupt (VBL). The display on your video monitor must be refreshed every sixtieth of a second to keep it from fading away. This is accomplished by scanning every row of dot positions from top to bottom with an electron beam. There is a slight delay when the beam reaches the right side of the screen and has to travel all the way back to the left side, but there is an even longer delay while it travels from the lower-right corner back to the upper-left. The video circuitry generates an electrical signal when this process begins and it is interpreted by the microprocessor as an interrupt.

Normally, if you make changes to the display without coordinating them with the VBL, you'll get some degree of screen flicker. This is because the change often takes place in the middle of a screen refresh and you get part of the old screen contents at the same time you get some of the new screen contents. However, if you wait until the VBL begins to change the screen contents, there will be no flicker. The length of the VBL is limited, so there's a limit to how many changes you can make.

Listing 4 installs a short machine language program that alternately inverses and restores a line of text. The animation is flicker-free because the program waits for the VBL to occur before making the changes.

SETTING TEXT SCREEN COLORS

The Control Panel offers a choice of colors for background, border and text, but you can also control these colors from a program. Register 49186 ($C022) holds the background and text colors. However, the border color shares register 49204 ($C034) with bits that control the GS clock. The machine language program simply changes the four border color bits without changing the clock control bits. To use this routine, you POKE the desired color into location 800 and perform a CALL 768.

The screen informs you that pressing Q will end the program. You are then prompted for each of the three colors. Enter a number from 0 to 15 at each prompt. Consult **Table 2** for the colors that correspond to these values. After you respond to the last prompt, the screen colors change, and the quit message is displayed again, followed by the first color prompt. Press Q to restore the default colors and quit.

CYCLECOLS (**Listing 6**) uses the same techniques to change all three colors at once, rapidly cycling through all 16 colors until you press a key. **Line 160** waits for the VBL before continuing. Because BASIC is so slow, the pause signaled by the VBL is over before the changes to the colors are actually made. However, this synchronization does keep the horizontal line in about the same place on the screen. The machine language program is the same as the one used in **Listing 5.**

SPECIAL EFFECTS

Listings 7, 8 and 9 produce special effects by rapidly changing the Color Registers without waiting for a VBL. Using this method, the area of the screen whose color is changed will appear with scrolling stripes of the various colors used. To stop any of these programs and return to a normal screen, press any key.

Listing 7 sets up a text screen with a solid block of inverse spaces in the middle. This effect is achieved by rapidly changing the contents of the text and background color registers.

Listing 8 puts stripes in the screen border. To change the delay between stripes, just POKE a different value into register 788. It is currently 128, but it can range from 1 to 255. In addition, you can change the colors in the border by POKEing different values (from 0 to 15) into registers 806, 807, 808 and 809.

Listing 9 accomplishes the same thing, only with the screen background and text color registers. The program is nearly identical to **Listing 8**, so you can make the same modifications. The color values are calculated by the formula:

16 * *CHAR* + *BACK*

where *CHAR* and *BACK* are values from 0 to 15 representing the desired colors.

ENTERING THE PROGRAMS

Listing 4 works on an enhanced IIe, IIc or IIGS. All the other programs require a IIGS. Enter the programs as follows:

1. Type in the program in Listing 1 and save it with the command:

   SAVE GS.MODIFIERS

2. Type in **Listing 2** and save it with the command:

   SAVE SPEED.TEST

3. Type in **Listing 3** and save it with the command:

   SAVE MONOCHROME

4. Type in **Listing 4** and save it with the command:

   SAVE BLINKY

5. Type in **Listing 5** and save it with the command:

   SAVE SET.COLORS

6. Type in **Listing 6** and save it with the command:

   SAVE CYCLECOLS

7. Type in **Listing 7** and save it with the command:

   SAVE ZAPPO

8. Type in **Listing 8** and save it with the command:

   SAVE BORDEAUX

9. Type in **Listing 9** and save it with the command:

   SAVE TEXTO

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

LISTING 1: GS.MODIFIERS

```
10  REM  **********************
20  REM  *    GS.MODIFIERS    *
30  REM  * BY JON C. THOMASON *
40  REM  * COPYRIGHT (C) 1987 *
50  REM  * BY MICROSPARC, INC *
60  REM  * CONCORD, MA  Ø1742 *
70  REM  **********************
80  HOME : PRINT "Press any key combination:"
90  GET KEY$:MOD =  PEEK (49189)
100  GOSUB 360: REM  Break into bits
110  IF BIT(7) THEN  PRINT "Open Apple-";
120  IF BIT(6) THEN  PRINT "Option-";
130  IF BIT(2) THEN  PRINT "Caps Lock-";
140  IF BIT(Ø) THEN  PRINT "Shift-";
150  IF BIT(1) THEN  PRINT "Control-";
160  IF KEY$ = "" THEN KEY = Ø: GOTO 180
170 KEY =  ASC (KEY$)
180  IF KEY < 33 THEN  GOSUB 45Ø: ON FF GOTO
     32Ø: GOTO 31Ø
190  IF BIT(4) GOTO 31Ø
200  IF KEY < 44 THEN SEARCH$ = "!#$%^&*()+" +
      CHR$ (34):TARGT$ = "13456789Ø='": GOSUB
     58Ø: IF FF THEN 32Ø
210  IF KEY < 58 OR KEY = 96 OR (KEY > 64 AND
     KEY < 94) GOTO 31Ø
220  IF KEY = 58 THEN KEY = 59: GOTO 31Ø
230  IF KEY = 59 OR KEY = 61 GOTO 31Ø
240  IF KEY < 64 THEN KEY = KEY - 16: GOTO 31
     Ø
250  IF KEY = 64 THEN KEY = 5Ø: GOTO 31Ø
255  IF KEY = 94 THEN KEY = 54: GOTO 31Ø
260  IF KEY = 95 THEN KEY = 45: GOTO 31Ø
270  IF KEY = 126 THEN KEY = 96: GOTO 31Ø
290  IF KEY = 127 THEN KEY$ = "Delete": GOTO
     32Ø
300 KEY = KEY - 32
310 KEY$ =  CHR$ (KEY)
320  PRINT KEY$;"  ";
330  IF BIT(4) THEN  PRINT "(Keypad)  ";
340  IF BIT(3) THEN  PRINT "REPEAT...";
350  PRINT : GOTO 9Ø
360  FOR X = Ø TO 7:BIT(X) = Ø: NEXT
370  IF MOD > 127 THEN BIT(7) = 1:MOD = MOD -
     128
380  IF MOD > 63 THEN BIT(6) = 1:MOD = MOD -
     64
390  IF MOD > 31 THEN BIT(5) = 1:MOD = MOD -
     32
400  IF MOD > 15 THEN BIT(4) = 1:MOD = MOD -
     16
410  IF MOD > 7 THEN BIT(3) = 1:MOD = MOD - 8
420  IF MOD > 3 THEN BIT(2) = 1:MOD = MOD - 4
430  IF MOD > 1 THEN BIT(1) = 1:MOD = MOD - 2
440 BIT(Ø) = MOD: RETURN
450 FF = Ø: IF BIT(1) THEN 55Ø
460  IF KEY = 8 THEN KEY$ = "Left Arrow": GOTO
     56Ø
470  IF KEY = 9 THEN KEY$ = "Tab": GOTO 56Ø
480  IF KEY = 1Ø THEN KEY$ = "Down Arrow": GOTO
     56Ø
490  IF KEY = 11 THEN KEY$ = "Up Arrow": GOTO
     56Ø
500  IF KEY = 13 THEN KEY$ =  MID$ ("Return E
     nter  ",BIT(4) * 7 + 1,7): GOTO 56Ø
510  IF KEY = 21 THEN KEY$ = "Right Arrow": GOTO
     56Ø
520  IF KEY = 24 THEN  IF BIT(4) THEN KEY$ =
     "Clear": GOTO 56Ø
530  IF KEY = 27 THEN KEY$ = "Escape": GOTO 5
     6Ø
540  IF KEY = 32 THEN KEY$ = "Space ": GOTO 5
     6Ø
550 KEY = KEY + 64: GOTO 57Ø
560 FF = 1
570  RETURN
580 FF = Ø: FOR I = 1 TO  LEN (SEARCH$)
590 A$ =  MID$ (SEARCH$,I,1)
600  IF KEY$ = A$ THEN KEY$ =  MID$ (TARGT$,I
     ,1):I =  LEN (SEARCH$):FF = 1
610  NEXT I
620  RETURN
```

END OF LISTING 1

```
                 KEY PERFECT 5.Ø
                     RUN ON
                  GS.MODIFIERS
 ======================================
     CODE-5.Ø    LINE# - LINE#    CODE-4.Ø
     --------    -------------    --------
     Ø3B13486      1Ø -   1ØØ       6B7F
     1F38AØF3     11Ø -   2ØØ       81A2
     927ECCDF     21Ø -   3ØØ       7455
     CE97AE1D     31Ø -   4ØØ       6E1C
     8981AD9E     41Ø -   5ØØ       8521
     9C1928FE     51Ø -   6ØØ       7564
     DFFBD5D1     61Ø -   62Ø       Ø4Ø3
     ED2F967D = PROGRAM TOTAL =     Ø618
```

LISTING 2: SPEED.TEST

```
10  REM  **********************
20  REM  * SPEED.TEST         *
30  REM  * BY JON THOMASON    *
40  REM  * COPYRIGHT (C) 1987 *
50  REM  * BY MICROSPARC, INC *
60  REM  * CONCORD, MA  Ø1742 *
70  REM  **********************
80 FAST =  PEEK (492Ø6) > 127: REM $CØ36
90  IF FAST THEN  PRINT "Fast mode."
100  IF  NOT FAST THEN  PRINT "Normal mode."
```

END OF LISTING 2

LISTING 3: MONOCHROME

```
10  REM  **********************
20  REM  * MONOCHROME         *
30  REM  * BY JON THOMASON    *
40  REM  * COPYRIGHT (C) 1987 *
50  REM  * BY MICROSPARC, INC *
60  REM  * CONCORD, MA  Ø1742 *
70  REM  **********************
80  HGR : HOME : PRINT  CHR$ (17);
90  FOR X = Ø TO 16Ø
100  HCOLOR= X -  INT (X / 6) * 6 + 1: HPLOT
     Ø,X TO 279,X
110  NEXT : POKE 49246,Ø: REM $CØ5E
120  VTAB 21: PRINT "Set Control Panel to Mon
     ochrome"
```

END OF LISTING 3

LISTING 4: BLINKY

```
10  REM  **********************
20  REM  * BLINKY             *
30  REM  * BY JON THOMASON    *
40  REM  * COPYRIGHT (C) 1987 *
50  REM  * BY MICROSPARC, INC *
60  REM  * CONCORD, MA  Ø1742 *
70  REM  **********************
80 A$ = "Written by Jon C. Thomason"
90  FOR X = 768 TO 8Ø2: READ Y: POKE X,Y: NEXT

100  PRINT  CHR$ (4)"PR#3": PRINT  CHR$ (17) CHR$
     (27)
110  TEXT : FOR X = .25 TO 1Ø
120  VTAB 1Ø: NORMAL : HTAB X * 4: PRINT "___
     _": REM  4 UNDERLINES
130  VTAB 12: INVERSE : HTAB X * 4: PRINT "LL
     LL": REM  4 L'S
140 S$ = S$ + "     ": NEXT : PRINT  CHR$ (24
     ): REM  5 SPACES
150  VTAB 11: CALL  - 868: HTAB 21 -  LEN (A$
     ) / 2: PRINT A$
160  NORMAL : VTAB 11: CALL 768: VTAB 2Ø
170  DATA 173,54,192,41,127,141,54,192,16Ø,39
     ,177,4Ø,73,128,145,4Ø,136
180  DATA 16,247,173,Ø,192,48,7,173,25,192,16
     ,251,48,233,44,16,192,96
```

END OF LISTING 4

```
                 KEY PERFECT 5.Ø
                     RUN ON
                     BLINKY
 ======================================
     CODE-5.Ø    LINE# - LINE#    CODE-4.Ø
     --------    -------------    --------
     62AØDA83      1Ø -   1ØØ       6C1B
     8B76EE3A     11Ø -   18Ø       8325
     ABB1F2FØ = PROGRAM TOTAL =     Ø2ØB
```

LISTING 5: SET.COLORS

```
10  REM  **********************
20  REM  * SET.COLORS         *
30  REM  * BY JON THOMASON    *
40  REM  * COPYRIGHT (C) 1987 *
50  REM  * BY MICROSPARC, INC *
60  REM  * CONCORD, MA  Ø1742 *
70  REM  **********************
80  HOME : FOR I = Ø TO 11: READ X: POKE 768 +
     I,X: NEXT
90  INVERSE : PRINT "ENTER Q AT ANY PROMPT TO
      QUIT": NORMAL
100  INPUT "BACKGROUND COLOR (Ø-15): ";X$:BAC
     K =  VAL (X$): ON X$ = "Q" OR X$ = "q" GOTO
     16Ø: IF BACK < Ø OR BACK > 15 OR X$ = ""
      THEN  PRINT  CHR$ (7): GOTO 1ØØ
110  INPUT "TEXT COLOR (Ø-15): ";X$:TXT =  VAL
     (X$): ON X$ = "Q" OR X$ = "q" GOTO 16Ø: IF
```

LISTING 5: SET.COLORS *(continued)*

```
     TXT < 0 OR TXT > 15 OR X$ = "" THEN  PRINT
      CHR$ (7): GOTO 110
120  INPUT "BORDER COLOR (0-15): ";BRDR$:BRDR
      =  VAL (BRDR$): ON X$ = "Q" OR X$ = "q"
      GOTO 160: IF BRDR < 0 OR BRDR > 15 OR B
     RDR$ = "" THEN  PRINT  CHR$ (7): GOTO 12
     0
130  POKE 49186,BACK + 16 * TXT: REM  $C022
140  POKE 800,BRDR: CALL 768: REM  $C034 ALSO
      CONTROLS CLOCK
150  GOTO 90
160  POKE 49186,6 + 16 * 15: POKE 800,6: CALL
     768: REM  RESTORE DEFAULT COLORS
170  END
180  DATA 173,52,192,41,240,13,32,3,141,52,19
     2,96
```

END OF LISTING 5

```
                KEY PERFECT 5.0
                    RUN ON
                  SET.COLORS
      ========================================
        CODE-5.0    LINE# - LINE#    CODE-4.0
        --------    -------------    --------
        5FD975FE       10 -   100      90E9
        1F2A66FA      110 -   180      A781
        C53BF414 = PROGRAM TOTAL =     02AA
```

LISTING 6: CYCLECOLS

```
10   REM  *********************
20   REM  * CYCLECOLS         *
30   REM  * BY JON THOMASON   *
40   REM  * COPYRIGHT (C) 1987 *
50   REM  * BY MICROSPARC, INC *
60   REM  * CONCORD, MA  01742 *
70   REM  *********************
80 BRDER = 0:BACK = 0:CHAR = 0: FOR I = 0 TO
     11: READ X: POKE 768 + I,X: NEXT
90   POKE 49186,BACK + 16 * CHAR: REM $C022
100  POKE 800,BACK: CALL 768: REM $C034 also
     controls clock
110 BRDER = BRDER + 1:BACK = BACK + 1:CHAR =
     CHAR + 1
120  IF BRDER > 15 THEN BRDER = BRDER - 16
130  IF BACK > 15 THEN BACK = BACK - 16
140  IF CHAR > 15 THEN CHAR = CHAR - 16
150  FOR X = 1 TO 100: NEXT
160  IF  PEEK (49177) < 128 THEN 160
170  IF  PEEK (49152) < 128 GOTO 90
180  POKE 49186,6 + 15 * 16: POKE 800,6: CALL
     768: REM  restore default colors
190  END
200  DATA 173,52,192,41,240,13,32,3,141,52,19
     2,96
```

END OF LISTING 6

```
                KEY PERFECT 5.0
                    RUN ON
                  CYCLECOLS
      ========================================
        CODE-5.0    LINE# - LINE#    CODE-4.0
        --------    -------------    --------
        2DB44A7B       10 -   100      877A
        8E02B9CE      110 -   200      7F27
        AD2462BE = PROGRAM TOTAL =     022A
```

LISTING 7: ZAPPO

```
10   REM  *********************
20   REM  * ZAPPO             *
30   REM  * BY JON THOMASON   *
40   REM  * COPYRIGHT (C) 1987 *
50   REM  * BY MICROSPARC, INC *
60   REM  * CONCORD, MA  01742 *
70   REM  *********************
80   HOME : PRINT  CHR$ (17);: INVERSE
90   FOR X = 10 TO 15: VTAB X: PRINT  TAB( 41)
100  NEXT : NORMAL
110  FOR X = 768 TO 790: READ Y
120  POKE X,Y: NEXT : CALL 768
130  DATA 173,34,192,72,24,105,17,141,34,192,
     174,0
140  DATA 192,16,245,44,16,192,104,141,34,192
     ,96
```

END OF LISTING 7

LISTING 8: BORDEAUX

```
10   REM  *********************
20   REM  * BORDEAUX          *
30   REM  * BY JON THOMASON   *
40   REM  * COPYRIGHT (C) 1987 *
50   REM  * BY MICROSPARC, INC *
60   REM  * CONCORD, MA  01742 *
70   REM  *********************
80   FOR X = 768 TO 810: READ Y
90   POKE X,Y: NEXT : CALL 768
100  REM  POKE 788,time delay (1-255)
110  REM  Color table starts at $325
120  DATA 173,52,192,72,172,37,3,185,38,3,141
     ,52,192,136
130  DATA 16,3,172,37,3,162,128,202,208,253,1
     74,0,192,16
140  DATA 234,104,141,52,192,44,16,192,96,4,0
     ,2,6,7,15
```

END OF LISTING 8

LISTING 9: TEXTO

```
10   REM  *********************
20   REM  * TEXTO             *
30   REM  * BY JON THOMASON   *
40   REM  * COPYRIGHT (C) 1987 *
50   REM  * BY MICROSPARC, INC *
60   REM  * CONCORD, MA  01742 *
70   REM  *********************
80   FOR X = 768 TO 810: READ Y
90   POKE X,Y: NEXT : CALL 768
100  REM  POKE 788,time delay (1-255)
110  REM  Color table starts at $325
120  DATA 173,34,192,72,172,37,3,185,38,3,141
     ,34,192,136,16
130  DATA 3,172,37,3,162,128,202,208,253,174,
     0,192,16,234,104
140  DATA 141,34,192,44,16,192,96,4,192,226,2
     14,247,255
```

END OF LISTING 9