HI-RES COLOR FLASH

Show off your graphics by adding an attention-grabbing visual effect

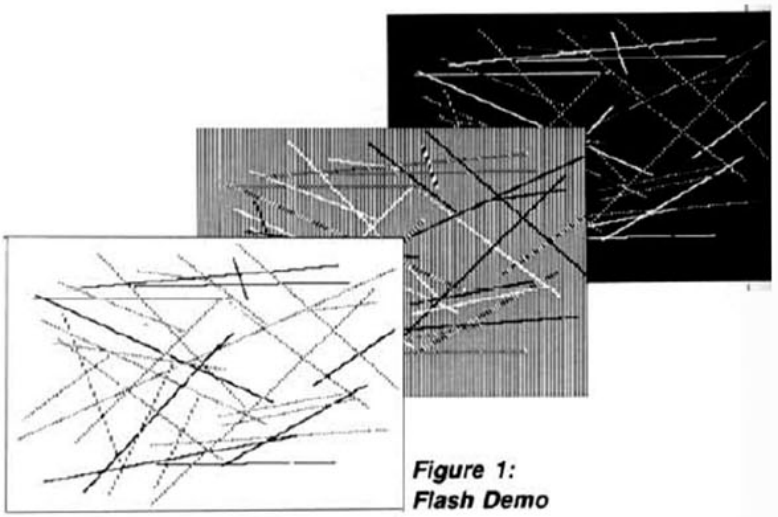

Figure 1: Flash Demo

In an early issue of *Nibble* (Vol. 3/No. 4), Todd Livesey showed us how to make the Hi-Res screen flash. His program produces some interesting effects by changing the color bit in each byte and thus inverting the Hi-Res screen. But after watching a computer game in which the screen not only inverted but changed from a black background to a colorful one, I set out to modify Todd's program so it too could do this.

USING THE PROGRAM

To see Flash in action, RUN the demo program (**Listing 3**). If you want to use FLASH for your own picture, forget the demo and do the following: type HGR, BLOAD the picture onto Hi-Res page 1 or page 2, and then type

```
BLOAD FLASH: POKE 6,color: CALL 768
```

Use HGR2 if you loaded your picture onto page 2. *Color* is a number from 0 to 7, corresponding to the Hi-Res colors. See **Figure 1** for results of the demo.

ENTERING THE PROGRAM

To enter the code, you can assemble it from **Listing 1** if you intend to relocate or change it, or you can enter the hex data in **Listing 2**, starting at $300, and then type

```
BSAVE FLASH,A$300,L$5C
```

Type in the Applesoft program in **Listing 3** and save it to disk with the command

```
SAVE FLASH.DEMO
```

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

HOW THE PROGRAM WORKS

As it stands, Todd's program is limited because of the arrangement of the Hi-Res screen. Because the byte value for any color besides black or white is different in odd and even columns, the results of Exclusive ORing (EORing) in an odd column must be different from EORing the same number in an even column. In order for my Hi-Res Color Flash to change black into color, it must distinguish between odd and even columns.

The numbers used to obtain different colors are stored in two tables at the end of the program, COLEVEN and COLODD. These tables are accessed once per flash using the Y register as an index. Y contains, on entry, the color number. Notice that for blacks and whites (colors 0, 3, 4, and 7) the two numbers are the same, while for the other colors they are different. Notice also that selecting zero as your color will produce no effect.

The program has two entry points: BASICENT (768, or $300) for BASIC, and MACHENT (770, or $302) for machine language. The only difference is that BASICENT loads the Y register with the color number, which was POKEd into location 6, while MACHENT assumes the number is already in the Y register.

On entry, the program loads the values as described before and stores them inside the next section of the program. This section loops through the entire screen, doing one even column byte and one odd column byte per loop.

At the machine language entry point, the program looks for the Hi-Res page number and the specified color. The page number is stored in memory location 230, or $E6. A 32, or $20, indicates page 1, while a 64, or $40, specifies page 2. This location is set correctly by HGR and HGR2. The color to flash is the number of the color to which a black zero background would be switched.

LISTING 1: HI.RES.FLASH Source Code

```
1 * HI.RES.FLASH Source Code
2 * By Phil Goetz
3 * Copyright(c) 1988
4 * MicroSPARC, Inc.
5 * Concord, MA 01742
6 *
7 * Merlin Pro Assembler
8 *
9           ORG   $300        ;CAN BE RE-ASSEMBLED ANYWHERE
10 BASICENT LDY   $6          ;MEMORY PAGE LOCATION
11 MACHENT  LDA   $E6
12          STA   INVERSE1+2
13          STA   STORE1+2
14          STA   INVERSE2+2
15          STA   STORE2+2
16          CLC
17          ADC   #$20
18          STA   LIMIT+1
19          LDA   COLEVEN,Y
20          STA   FLASH1+1
21          LDA   COLODD,Y
22          STA   FLASH2+1
23          LDY   #0
24 FLASH1   LDA   #0
25 INVERSE1 EOR   $0000,Y
26 STORE1   STA   $0000,Y
27          INY
28 FLASH2   LDA   #0
29 INVERSE2 EOR   $0000,Y
30 STORE2   STA   $0000,Y
31          INY
32          BNE   FLASH1
33          INC   INVERSE1+2
34          INC   STORE1+2
35          INC   INVERSE2+2
36          INC   STORE2+2
37          LDA   INVERSE1+2
38 LIMIT    CMP   #0
39          BNE   FLASH1
40          RTS
41 COLEVEN  DFB   $0,$55,$2A,$7F,$80,$D5,$AA,$FF
42 COLODD   DFB   $0,$2A,$55,$7F,$80,$AA,$D5,$FF
43 *        END
```

END OF LISTING 1

LISTING 2: FLASH

```
START: 300                    LENGTH:5C
C4 0300:A4 06 A5 E6 8D 28 03 8D
30 0308:2B 03 8D 31 03 8D 34 03
93 0310:18 69 20 8D 48 03 B9 4C
AD 0318:03 8D 25 03 B9 54 03 8D
BD 0320:2E 03 A0 00 A9 00 59 00
DB 0328:00 99 00 00 C8 A9 00 59
C1 0330:00 00 99 00 00 C8 D0 EC
94 0338:EE 28 03 EE 2B 03 EE 31
13 0340:03 EE 34 03 AD 28 03 C9
10 0348:00 D0 D9 60 00 55 2A 7F
DE 0350:80 D5 AA FF 00 2A 55 7F
4E 0358:80 AA D5 FF

TOTAL: 0A20
```

END OF LISTING 2

LISTING 3: FLASH.DEMO

```
37 | 10  REM **********************
C0 | 20  REM * FLASH.DEMO         *
B9 | 30  REM * BY PHIL GOETZ      *
AE | 40  REM * COPYRIGHT(C) 1988  *
CB | 50  REM * MICROSPARC, INC.   *
24 | 60  REM * CONCORD, MA 01742  *
45 | 70  REM **********************
47 | 80  ONERR  GOTO 170
7C | 90  PRINT  CHR$ (4)"BLOAD FLASH,A$300"
09 | 100  HGR
84 | 110  FOR I = 1 TO 5: FOR C = 1 TO 7: HCOLOR= C
CC | 120  HPLOT  RND (1) * 279, RND (1) * 191 TO  RN
       D (1) * 279, RND (1) * 191
42 | 130  NEXT : NEXT
A1 | 140  FOR I = 1 TO 7: POKE 6,I: CALL 768: GOSUB
       160: NEXT
68 | 150  TEXT : HOME : VTAB 23: END
32 | 160  HOME : VTAB 23: PRINT "PRESS RETURN TO CON
       TINUE ";: POKE  - 16368,0: GET A$: RETURN
19 | 170  PRINT "ERROR WHILE LOADING 'FLASH'": END
```

TOTAL: F667

END OF LISTING 3