Tips 'N Techniques

DOS 3.3 | ProDOS

HI-RES SPECIAL EFFECTS

These two short machine language routines will let you reverse the colors in a Hi-Res picture or flip it to get a mirror image. An Applesoft demonstration program shows you how it's done.

by Rod MacKenzie
5755 SW 57th Terrace
Miami, FL 33143

Recent advertisements for special effects graphics packages have inspired me to write two assembly language programs that you may wish to add to your Hi-Res utility program collection.

The first program, POS.NEG, performs a color reversal on the entire page 1 Hi-Res screen ($2000-3FFF). In the case of black and white images, the black areas change to white and the white areas change to black, effectively producing a negative image from a positive image or vice versa. In the case of color pictures, the complement of the existing color is produced; which is, to say the least, a picture of a different color.

The second program, MIRROR.IMAGE, transforms the page 1 Hi-Res screen into its mirror image counterpart. If you look at the reflection of your page 1 picture in a mirror, that is how it will appear after executing this program. The bit manipulations occurring within MIRROR.IMAGE also produce some interesting color changes in the transformed picture, because of the way color is mapped onto the Hi-Res screen. The color effects produced by MIRROR.IMAGE will not be quite as noticeable if you are using a black and white display screen.

A third program, SPECIAL.EFFECTS, demonstrates the use of these routines from within a BASIC program. SPECIAL.EFFECTS also allows you to observe the Hi-Res images as they change, if you enter your menu selections while looking at the Hi-Res page 1 image.

Entering the Programs

If you don't have an assembler you will have to enter the Monitor and type in the hex codes directly. (For further information, see the "Welcome to New *Nibble* Readers" at the beginning of this issue.) The POS.NEG program (**Listing 1**) is so short that this may be the preferred method even if you have an assembler. MIRROR.IMAGE (**Listing 2**) is somewhat longer, however, so take care when typing it in.

To save the programs, use the following commands:

BSAVE POS.NEG,A$610E,L$4C

BSAVE MIRROR.IMAGE,A$6000,L$10E

POS.NEG is a relocatable program and will therefore execute properly at other locations in memory without modification. MIRROR.IMAGE is nonrelocatable because of two JSR FLIP commands. If you decide to relocate MIRROR.IMAGE, the address of the FLIP subroutine will change ($60A6 in **Listing 2**). It is a simple exercise to locate the new address of the FLIP subroutine and alter the two JSR FLIP commands to reflect this change.

Using the Programs

POS.NEG and MIRROR.IMAGE may be called from a BASIC program, directly from the Monitor, or from within an assembly language program.

It is assumed that you have drawn or have otherwise placed a Hi-Res image on page 1 of the Hi-Res screen. Many Hi-Res arcade-type games leave pictures in this area of memory, which you can use if you can exit or otherwise break out of the game without having to turn the power off. (Pressing the <RESET> key is one possibility.) You may then be able to save the picture remaining with:

BSAVE PICTURE,A$2000,L$1FF8

Running From the Monitor:

CALL -151 <RETURN>	Enter the Monitor.
BLOAD PICTURE,A$2000 <RETURN>	Omit this step if your picture is already in memory.
BLOAD MIRROR.IMAGE ,A$6000 <RETURN>	
BLOAD POS.NEG,A$610E <RETURN>	
CO57 <RETURN>	Hi-Res graphics switch.
CO52 <RETURN>	Full page switch.
CO54 <RETURN>	Page 1 switch.
CO50 <RETURN>	Display graphics switch. You should now be looking at your picture. From now on you won't be able to see what you type...so type carefully.
610EG <RETURN>	Run POS.NEG.
610EG <RETURN>	Run it again.
6000G <RETURN>	Run MIRROR.IMAGE.
6000G <RETURN>	Run it again.
CO51 <RETURN>	Text switch...return to text mode.
3DOG <RETURN>	Exit the Monitor.

Running From BASIC:

Type in and save the BASIC program SPECIAL.EFFECTS (**Listing 3**) on the same disk that you saved the machine language routines. SPECIAL.EFFECTS assumes that you have BSAVEd these routines at the specified addresses, and with the same names as previously given. SPECIAL.EFFECTS gives you the option of loading a picture which you have previously saved to disk by selecting option 1 in response to the prompt. Menu selections may be entered while in graphics or text mode.

The Hi-Res Screen

The Hi-Res page 1 screen is really composed of 64 consecutive sequences of 128 bytes each. The eight bytes at the end of each sequence are unused, leaving the 120 bytes that are displayed on the screen. However, the 120 bytes are further subdivided into consecutive groups of 40 bytes, with each group of 40 bytes forming one screen line. There are therefore three screen lines within each 128-byte sequence.

Program Logic

The POS.NEG program simply complements (EOR #$7F) the 120 bytes beginning at $2000. It then adds eight bytes to the calculated screen address, in order to pass by the eight unused bytes at the end of each group of 120 bytes. This process is repeated until the end condition is detected (screen address = $3FF8).

The MIRROR.IMAGE logic is relatively complex. MIRROR.IMAGE uses BASEL and BASEH to hold the beginning address of each screen line as it processes each 128-byte sequence. The routine CALC uses BASEL and BASEH to calculate the start and end of each screen line (STARTL-STARTH and ENDL-ENDH).

The screen line is then processed by the routine labelled SWITCH. The SWITCH routine exchanges the first byte with the last byte, the second byte with the second to last byte, etc., until the middle two bytes have been exchanged.

The variable CNT1 signals when the exchanges have been completed. However, during the exchange process, the FLIP subroutine is called. This subroutine flips each byte 180 degrees (for example, 00010101 becomes 10101000) in order to preserve the sequencing of bits between adjoining bytes. The FLIP subroutine also shifts each byte right one bit, in order to prevent the plotting of nonplotting color bits. The variable CNT2 signals when three screen lines have been processed, so that the base address (BASEL-BASEH) can be adjusted to pass by the eight unused bytes.

If you change the LSR A instruction in the FLIP subroutine of MIRROR.IMAGE to ROR A, you will obtain a different set of color changes in the transformed picture. If you do not have an assembler, this change can be made from the Monitor by changing 610C:4A to 610C:6A.

In Conclusion

I hope that you will have as much fun as I have had experimenting with these routines on various screen images.

LISTING 1: POS.NEG

```
SOURCE FILE: POS.NEG.TEXT
0000:                1 ********************************
0000:                2 *                              *
0000:                3 *           POS.NEG            *
0000:                4 *                              *
0000:                5 *      BY ROD MACKENZIE        *
0000:                6 *      COPYRIGHT (C) 1984      *
0000:                7 *      MICROSPARC, INC.        *
0000:                8 *      LINCOLN, MA 01773       *
0000:                9 *                              *
0000:               10 *      ASSEMBLER: DOS TOOL KIT *
0000:               11 *                              *
0000:               12 * THIS PROGRAM COMPLEMENTS     *
0000:               13 * HI-RES PAGE 1 SCREEN BYTES   *
0000:               14 * PRODUCING NEGATIVE IMAGES    *
0000:               15 * FROM POSITIVE ONES AND       *
0000:               16 * VICE VERSA.                  *
0000:               17 *                              *
0000:               18 ********************************
----- NEXT OBJECT FILE NAME IS POS.NEG
610E:               19          ORG  $610E
610E:               20 ;
610E:               21 ;
0000:               22 ADDRL    EQU  $00
0001:               23 ADDRH    EQU  $01
0002:               24 CNT      EQU  $02
610E:               25 ;
610E:               26 *  INITIALIZE  *
610E:A9 00          27          LDA  #$00
6110:85 00          28          STA  ADDRL
6112:85 02          29          STA  CNT
6114:A9 20          30          LDA  #$20
6116:85 01          31          STA  ADDRH
6118:               32 ;
6118:               33 * GET A SCREEN BYTE     *
6118:               34 * COMPLEMENT & REPLACE  *
6118:A2 00          35 L1:      LDX  #$00
611A:A1 00          36          LDA  (ADDRL,X)
611C:49 7F          37          EOR  #$7F
611E:81 00          38          STA  (ADDRL,X)
6120:               39 ;
6120:               40 * ADD 1 TO SCREEN ADDRESS *
6120:18             41          CLC
6121:A5 00          42          LDA  ADDRL
6123:69 01          43          ADC  #$01
6125:85 00          44          STA  ADDRL
6127:A5 01          45          LDA  ADDRH
6129:69 00          46          ADC  #$00
612B:85 01          47          STA  ADDRH
612D:               48 ;
612D:               49 * FINISHED IF SCREEN *
612D:               50 * ADDRESS = $3FF8     *
612D:A5 00          51          LDA  ADDRL
612F:C9 F8          52          CMP  #$F8
6131:F0 03          53          BEQ  L2
6133:38             54          SEC
6134:B0 06          55          BCS  L3
6136:               56 ;
6136:A5 01          57 L2:      LDA  ADDRH
6138:C9 3F          58          CMP  #$3F
613A:F0 1D          59          BEQ  L5
613C:               60 ;
613C:               61 * 120 BYTES YET ? *
613C:E6 02          62 L3:      INC  CNT
613E:C9 78          63          CMP  #$78
6140:F0 03          64          BEQ  L4
6142:38             65          SEC
6143:B0 D3          66          BCS  L1
6145:               67 ;
6145:               68 * RESET BYTE COUNTER *
6145:A9 00          69 L4:      LDA  #$00
6147:85 02          70          STA  CNT
6149:               71 ;
6149:               72 * ADD 8 TO ADDRESS TO PASS *
6149:               73 * BY THE 8 UNUSED BYTES     *
6149:18             74          CLC
614A:A5 00          75          LDA  ADDRL
614C:69 08          76          ADC  #$08
614E:85 00          77          STA  ADDRL
6150:A5 01          78          LDA  ADDRH
6152:69 00          79          ADC  #$00
6154:85 01          80          STA  ADDRH
6156:38             81          SEC
6157:B0 BF          82          BCS  L1
6159:               83 ;
6159:               84 * FINISHED *
6159:60             85 L5:      RTS

--- SUCCESSFUL ASSEMBLY: NO ERRORS
```

LISTING 2: MIRROR.IMAGE

```
SOURCE FILE: MIRROR.IMAGE.TEXT
0000:                 1 ****************************
0000:                 2 *                          *
0000:                 3 *       MIRROR.IMAGE       *
0000:                 4 *                          *
0000:                 5 *     BY ROD MACKENZIE     *
0000:                 6 *     COPYRIGHT (C) 1984   *
0000:                 7 *     MICROSPARC, INC.     *
0000:                 8 *     LINCOLN, MA 01773    *
0000:                 9 *                          *
0000:                10 *     ASSEMBLER: DOS TOOL KIT *
0000:                11 *                          *
0000:                12 * THIS PROGRAM CONSTRUCTS THE *
0000:                13 * MIRROR IMAGE OF A PICTURE  *
0000:                14 * STORED ON HI-RES PAGE ONE. *
0000:                15 * THE RESULTING IMAGE WILL   *
0000:                16 * REPLACE THE ORIGINAL.      *
0000:                17 *                          *
0000:                18 ****************************
----- NEXT OBJECT FILE NAME IS MIRROR.IMAGE
6000:                19          ORG  $6000
6000:                20 ;
0000:                21 STARTL   EQU  $00
0001:                22 STARTH   EQU  $01
0002:                23 ENDL     EQU  $02
0003:                24 ENDH     EQU  $03
0004:                25 BASEL    EQU  $04
0005:                26 BASEH    EQU  $05
0006:                27 CNT1     EQU  $06
0007:                28 CNT2     EQU  $07
0010:                29 TEMP     EQU  $10
6000:                30 ;
6000:                31 * SET UP TABLE FOR *
6000:                32 * SUBRTN = FLIP    *
6000:A9 01           33          LDA  #01
6002:85 08           34          STA  $08
6004:A9 02           35          LDA  #$02
6006:85 09           36          STA  $09
6008:A9 04           37          LDA  #$04
600A:85 0A           38          STA  $0A
600C:A9 08           39          LDA  #$08
600E:85 0B           40          STA  $0B
6010:A9 10           41          LDA  #$10
6012:85 0C           42          STA  $0C
6014:A9 20           43          LDA  #$20
6016:85 0D           44          STA  $0D
6018:A9 40           45          LDA  #$40
601A:85 0E           46          STA  $0E
601C:A9 80           47          LDA  #$80
601E:85 0F           48          STA  $0F
6020:                49 ;
6020:                50 * INITIALIZE *
6020:A9 00           51          LDA  #$00
6022:85 04           52          STA  BASEL
6024:A9 20           53          LDA  #$20
6026:85 05           54          STA  BASEH
6028:A9 14           55          LDA  #$14
602A:85 06           56          STA  CNT1
602C:A9 03           57          LDA  #$03
602E:85 07           58          STA  CNT2
6030:                59 ;
6030:                60 * CALCULATE START AND END *
6030:                61 * FROM BASEL AND BASEH    *
6030:A5 04           62 CALC     LDA  BASEL
6032:85 00           63          STA  STARTL
6034:A5 05           64          LDA  BASEH
6036:85 01           65          STA  STARTH
6038:18              66          CLC
6039:A5 04           67          LDA  BASEL
603B:69 27           68          ADC  #$27
603D:85 02           69          STA  ENDL
603F:A5 05           70          LDA  BASEH
6041:69 00           71          ADC  #$00
6043:85 03           72          STA  ENDH
6045:                73 ;
6045:                74 * SWITCH AND FLIP BYTES *
6045:A2 00           75 SWITCH   LDX  #$00
6047:A1 00           76          LDA  (STARTL,X)
6049:20 A6 60        77          JSR  FLIP
604C:48              78          PHA
604D:A1 02           79          LDA  (ENDL,X)
604F:20 A6 60        80          JSR  FLIP
6052:81 00           81          STA  (STARTL,X)
6054:68              82          PLA
6055:81 02           83          STA  (ENDL,X)
6057:                84 ;
6057:                85 * ADD 1 TO START *
6057:18              86          CLC
6058:A5 00           87          LDA  STARTL
605A:69 01           88          ADC  #$01
605C:85 00           89          STA  STARTL
605E:A5 01           90          LDA  STARTH
6060:69 00           91          ADC  #$00
6062:85 01           92          STA  STARTH
6064:                93 ;
6064:                94 * SUBTRACT 1 FROM END *
6064:38              95          SEC
6065:A5 02           96          LDA  ENDL
6067:E9 01           97          SBC  #$01
6069:85 02           98          STA  ENDL
606B:A5 03           99          LDA  ENDH
606D:E9 00          100          SBC  #$00
606F:85 03          101          STA  ENDH
6071:18             102          CLC
6072:               103 ;
6072:               104 * CNT1=CNT1 - 1 *
6072:C6 06          105          DEC  CNT1
6074:D0 CF          106          BNE  SWITCH
6076:               107 ;
6076:               108 * RESET CNT1=20 *
6076:A9 14          109          LDA  #$14
6078:85 06          110          STA  CNT1
607A:               111 ;
607A:               112 * CNT2=CNT2 - 1 *
607A:C6 07          113          DEC  CNT2
607C:D0 11          114          BNE  ADJBASE
607E:               115 ;
607E:               116 * SKIP THE NEXT 8 BYTES *
607E:18             117          CLC
607F:A5 04          118          LDA  BASEL
6081:69 08          119          ADC  #$08
6083:85 04          120          STA  BASEL
6085:A5 05          121          LDA  BASEH
6087:69 00          122          ADC  #$00
6089:85 05          123          STA  BASEH
608B:               124 ;
608B:               125 * RESET CNT2 *
608B:A9 03          126          LDA  #$03
608D:85 07          127          STA  CNT2
608F:               128 ;
608F:               129 * ADJUST BASE TO   *
608F:               130 * NEXT SCREEN LINE *
608F:18             131 ADJBASE CLC
6090:A5 04          132          LDA  BASEL
6092:69 28          133          ADC  #$28
6094:85 04          134          STA  BASEL
6096:A5 05          135          LDA  BASEH
6098:69 00          136          ADC  #$00
609A:85 05          137          STA  BASEH
609C:               138 ;
609C:               139 * FINISHED YET ? *
609C:A5 05          140          LDA  BASEH
609E:C9 40          141          CMP  #$40
60A0:F0 03          142          BEQ  END
60A2:38             143          SEC
60A3:B0 BB          144          BCS  CALC
60A5:               145 ;
60A5:               146 * THE END *
60A5:60             147 END      RTS
60A6:               148 ;
60A6:               149 * FLIP A BYTE 180 DEGREES *
60A6:               150 * AND THEN SHIFT IT RIGHT *
60A6:A0 00          151 FLIP     LDY  #$00
60A8:84 10          152          STY  TEMP
60AA:24 08          153 BIT0     BIT  $08
60AC:F0 08          154          BEQ  BIT1
60AE:48             155          PHA
60AF:A5 10          156          LDA  TEMP
60B1:09 80          157          ORA  #$80
60B3:85 10          158          STA  TEMP
60B5:68             159          PLA
60B6:24 09          160 BIT1     BIT  $09
60B8:F0 08          161          BEQ  BIT2
60BA:48             162          PHA
60BB:A5 10          163          LDA  TEMP
60BD:09 40          164          ORA  #$40
60BF:85 10          165          STA  TEMP
60C1:68             166          PLA
60C2:24 0A          167 BIT2     BIT  $0A
60C4:F0 08          168          BEQ  BIT3
60C6:48             169          PHA
60C7:A5 10          170          LDA  TEMP
60C9:09 20          171          ORA  #$20
60CB:85 10          172          STA  TEMP
60CD:68             173          PLA
60CE:24 0B          174 BIT3     BIT  $0B
60D0:F0 08          175          BEQ  BIT4
60D2:48             176          PHA
60D3:A5 10          177          LDA  TEMP
60D5:09 10          178          ORA  #$10
60D7:85 10          179          STA  TEMP
60D9:68             180          PLA
60DA:24 0C          181 BIT4     BIT  $0C
60DC:F0 08          182          BEQ  BIT5
60DE:48             183          PHA
```

```
60DF:A5 10      184          LDA  TEMP
60E1:09 08      185          ORA  #$08
60E3:85 10      186          STA  TEMP
60E5:68         187          PLA
60E6:24 0D      188 BIT5     BIT  $0D
60E8:F0 08      189          BEQ  BIT6
60EA:48         190          PHA
60EB:A5 10      191          LDA  TEMP
60ED:09 04      192          ORA  #$04
60EF:85 10      193          STA  TEMP
60F1:68         194          PLA
60F2:24 0E      195 BIT6     BIT  $0E
60F4:F0 08      196          BEQ  BIT7
60F6:48         197          PHA
60F7:A5 10      198          LDA  TEMP
60F9:09 02      199          ORA  #$02
60FB:85 10      200          STA  TEMP
60FD:68         201          PLA
60FE:24 0F      202 BIT7     BIT  $0F
6100:F0 08      203          BEQ  FIN
6102:48         204          PHA
6103:A5 10      205          LDA  TEMP
6105:09 01      206          ORA  #$01
6107:85 10      207          STA  TEMP
6109:68         208          PLA
610A:A5 10      209 FIN      LDA  TEMP
610C:4A         210          LSR  A
610D:60         211          RTS

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
KEY PERFECT 4.0
RUN ON
MIRROR.IMAGE
============================
CODE      ADDR# - ADDR#
------    -------------
25B5      6000 - 604F
2A46      6050 - 609F
2840      60A0 - 60EF
0F98      60F0 - 610D
PROGRAM CHECK IS : 010E
```

```
CHECK CODE 3.0

ON: MIRROR.IMAGE
TYPE: B

LENGTH: 010E
CHECKSUM: 73
```

LISTING 3: SPECIAL.EFFECTS

```
10  REM   **********************
20  REM   *   SPECIAL.EFFECTS   *
30  REM   *  BY ROD MACKENZIE   *
40  REM   * COPYRIGHT (C) 1984 *
50  REM   * BY MICROSPARC, INC *
60  REM   * LINCOLN, MA. 01773 *
70  REM   **********************
80  REM  LOAD THE ROUTINES
90  HOME
100  VTAB 12: HTAB 13: FLASH : PRINT "LOADING
     ";: NORMAL : PRINT " ROUTINES": VTAB 22:
      PRINT "** COPYRIGHT 1984 BY MICROSPARC,
      INC. **"
110  PRINT  CHR$ (4);"BLOAD MIRROR.IMAGE"
120  PRINT  CHR$ (4);"BLOAD POS.NEG"
130  REM  HEADING
140  DEF  FN A(X) =  INT ((40 -  LEN (A$)) /
     2)
150  HOME
160  FOR I = 1 TO 2: READ A$: GOSUB 550: NEXT

170  VTAB 6
180  REM  INSTRUCTIONS
190  FOR I = 1 TO 5
200  READ A$: PRINT : PRINT A$
210  NEXT
220  RESTORE
230 A$ = "HIT SPACE BAR TO EXIT GRAPHICS"
240  VTAB 20: GOSUB 550
250  VTAB 22:A$ = "<< ENTER SELECTION PLEASE>
     >"
260  CALL  - 958: GOSUB 550
270  REM  OBTAIN AND VERIFY SELECTION
280  PRINT "?";: GET B$:B =  VAL (B$)
290  IF  ASC (B$) = 32 THEN  TEXT : GOTO 130
300  IF B < 1 OR B > 5 THEN  PRINT  CHR$ (7):
      GOTO 250
310  ON B GOTO 320,380,410,440,470
320  REM  LOAD A PICTURE
330  TEXT : HOME
340  VTAB 12: PRINT "NAME OF PICTURE:";: INPUT
     PIC$
350  HOME : VTAB 12: FLASH : PRINT "LOADING";
     : NORMAL : PRINT "  ";: PRINT PIC$
360  PRINT  CHR$ (4);"BLOAD";PIC$;",A$2000"
370  GOTO 130
380  REM  VIEW PAGE 1
390  POKE  - 16297,0: POKE  - 16300,0: POKE  -
     16302,0: POKE  - 16304,0
400  GOTO 130
410  REM  COMPLEMENT THE PICTURE
420  CALL 24846: REM  ADDRESS OF POS-NEG = $6
     10E
430  GOTO 130
440  REM  MIRROR THE PICTURE
450  CALL 24576: REM  ADDRESS OF MIRROR-IMAGE
      = $6000
460  GOTO 130
470  TEXT : HOME : END
480  DATA  "HI-RES SPECIAL EFFECTS"
490  DATA   "----------------------": REM  22
      DASHES
500  DATA  "1). LOAD A BINARY PICTURE INTO PA
     GE 1"
510  DATA  "2). VIEW THE PAGE 1 PICTURE"
520  DATA  "3). COMPLEMENT THE PAGE 1 PICTURE
     "
530  DATA  "4). MIRROR THE PAGE 1 PICTURE"
540  DATA   "5). EXIT PROGRAM"
550  REM  SUBRTN. TO PRINT A$
560  PRINT  TAB(  FN A(X));A$
570  RETURN
```