HI RES FILL/REVERSE

Hi-Res Fill/Reverse

by Garry G. Kiziak
2381 Duncaster Drive
Burlington, ONT
Canada L7P 3V9

FAST FILL

This first routine is a very simple one that has many potential applications. Its simplicity lies in the fact that it does exactly one thing. However, it does that one thing very well, since it is written in machine language and hence is very fast. What does it do? **It quickly fills in any Hi-Res rectangle that you specify, with the color that you specify.**

To specify the rectangle, you must give the coordinates of the top left hand corner and the coordinates of the bottom right hand corner **(Figure 1.)**. These are the same coordinates that one would specify in an HPLOT command.

The color is specified by giving a number between 0 and 8 (just as in the HCOLOR command). The syntax of the CALLing statement is as follows:

CALL FILL,COLR,LC,TR,RC,BR,SP

where **FILL** = **768 (the starting address of the routine).**

COLR = **the number that specifies the color.**

LC,TR = **the coordinates of the top left corner of the rectangle.**

RC,BR = **the coordinates of the bottom right corner of the rectangle.**

SP = **the number of spaces to be left between each horizontal line.**

The reason for requiring the last quantity, SP, needs some clarification. **The routine fills in the rectangle by drawing horizontal lines, in the specified color, from the left column (LC) to the right column (RC) and from the top row (TR) to the bottom row (BR).** It also allows the possibility of leaving spaces between each line drawn. The variable, SP, specifies the number of spaces to be left. If SP = 0, then the rectangle will be filled in with a solid color. If SP = 1, then every other line will be drawn in the rectangle. If SP = 2, then every third line will be drawn, and so on.

TESTING FILL

Listing 2, is a short BASIC program that will allow you to experiment with filling in various rectangles on the HI RES screen. Type this program in, SAVE it, and then RUN it. In response to the prompts, try entering the following sequence of values:

The resulting picture should suggest one way in which this routine can be used; namely drawing 'BAR GRAPHS'. Notice that the 6th Box has wide strips of color and the 7th Box is multi-colored. This is accomplished by drawing several boxes on top of each other with different colors and/or slightly different sizes.

Listing 3, is another BASIC program to show how this routine can be used to 'jazz' up some of your displays (Shades of RASTER BLASTER). Nothing more will be said about this program. Simply type it in and RUN it to see what it does. The ways in which you can use this routine are clearly limited only by your imagination.

HOW FILL WORKS

Let's go through the assembler listing (Listing 1.) to explain how the routine works.

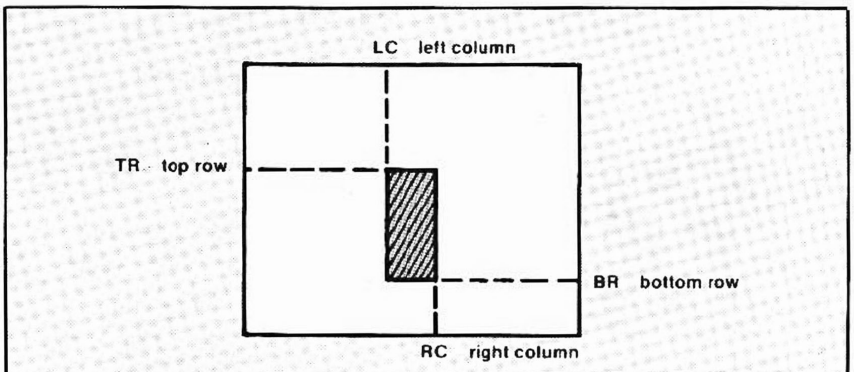

FIGURE 1

	COLOR	NUMBER OF SPACES	LEFT COLUMN	TOP ROW	RIGHT COLUMN	BOTTOM ROW
1st Box	1	0	0	100	20	158
2nd Box	2	1	30	100	50	158
3rd Box	3	2	60	100	80	158
4th Box	5	3	90	100	110	158
5th Box	6	4	120	100	140	158
6th Box	1	6	150	98	170	158
6th Box	1	6	150	99	170	158
6th Box	1	6	150	100	170	158
7th Box	1	5	180	96	200	158
7th Box	2	5	180	97	200	158
7th Box	3	5	180	98	200	158
7th Box	5	5	180	99	200	158
7th Box	6	5	180	100	200	158

Lines 11 to 34 get the various parameters and store them in temporary locations. Several data gathering routines from the Applesoft BASIC interpreter are used here, so perhaps a couple of comments about each one is in order.

COMBYTE: This routine checks for a comma at TXTPTR and evaluates the formula immediately after it. The result, which must fit into a single byte, is preserved in the X-register. The call to the COMBYTE routine in line 11 thus gets the number corresponding to the color of the rectangle. The call in line 32 gets the number corresponding to the number of spaces between the lines.

HCOLOR: This routine is the equivalent of the HCOLOR command in BASIC. The Hi-Res color must be stored in the X-register when this routine is called.

CHKCOM: This routine simply checks for a comma at TXTPTR and prints an error message if one is not there.

HFNS: This is a routine used by the BASIC interpreter to get the coordinates of a point required in an HPLOT command (e.g. HPLOT X1,Y1).

The value for Y1 must be between 0 and 191 and hence can fit into a single byte. When the HFNS routine returns, this value is stored in the accumulator. **The value for X1 can be between 0 and 279. Two bytes are necessary here since the value can be larger than 255.** When the HFNS routine returns, the low order byte of this value is in the X-register and the high order byte is in the Y-register. The first call to this routine (in line 17) gets the coordinates of the top left hand corner of the rectangle while the second call (line 25) gets the coordinates of the bottom right hand corner.

HPLOT: This routine calculates the proper screen address of the point whose coordinates are given in the accumulator (y-coordinate) and the X and Y registers (x-coordinate), and then plots that point on the Hi-Res screen in the color specified.

HLIN: This routine draws a straight line from the last point plotted to the point whose coordinates are given in the Y-register (y-coordinate) and the accumulator and X-register (x-coordinate), again in the specified color.

Lines 38 to 54 fill in the rectangle with lines 38 to 45 drawing the horizontal lines and line 48 adding the required number of spaces.

LOADING FILL

You can enter FILL directly into memory by typing CALL -151 to enter the Monitor. Then begin typing in the hex code from the listing as follows:

300:20 4C E7 20 F0 F6 20 BE etc.

When you have completed entering the hex code, type CTRL C to reenter Basic and then type **BSAVE FILL.OBJ0, A$300, L$57.**

For more instructions on entering hex code into memory, see the Letters section of this issue.

HI-RES REVERSE

This routine is similar in some respects to the FILL routine. **Its purpose is to reverse a specified rectangle on the Hi-Res screen (again either page).**

To specify the rectangle, the coordinates of the top left-hand corner and the bottom right-hand corner must be given. This time however, the x-coordinate of the points must be between 0 and 39 (instead of 0 and 279) — just as if we were on the text screen. The y-coordinate can still be between 0 and 191.

The CALLing statement has the following syntax:

CALL REVERSE,LC,TR,RC,BR

where **REVERSE** = **864 (the starting address of the routine).**
LC,TR = **the coordinates of the top left corner of the rectangle.**
RC,BR = **the coordinates of the bottom right corner of the rectangle.**

This routine **(Listing 4.)** is very simple. In fact, it is done in a single line (line 39). By 'Exclusive Oring' a byte with $FF (which is **11111111** in binary) you can change each bit in that byte from 0 to 1 and vice versa (including the bit that determines the color). The only other change required was in the routine that calculates the base address of a given screen line. Here we make use of the HPOSN routine.

Applications for this routine again depend only on your imagination. On the PET computer I have used a similar routine to create the effect of an explosion, simply by instantly reversing the screen several times. This is not directly possible with the HI-RES REVERSE routine because it is too slow (after all, you must alter 8K of memory on the Hi-Res screen, as opposed to 1K on the text screen). It is possible however, to use this routine and a little bit of trickery to achieve the same effect. **Listing 5.** is a program that does just that.

LOADING REVERSE

First type in the M/L routine and save it to disk with the command **BSAVE REVERSE HI-RES.OBJ0,A$360,LS2F**. Also put a Hi-Res picture on your disk and call it SPACEPIC, or else change the name in line 30 of Listing 5, accordingly.

TESTING REVERSE

When the program is run, you will see your Hi-Res picture appear on the screen. This picture was loaded into page 1, but then moved to page 2 (line 40) and then reversed (line 50). You won't see this since you are still looking at page 1. Then the program prompts you for input. Again you won't see the prompt, because it is on the text screen. Type in any number (a small one) and the program will reverse the screen that many times. Then it goes back for another input. This way you can experiment to see how many times the screen should be reversed in order to achieve the effect that you want. You can also experiment with the timing in lines 100 and 120 if you wish.

Finally, **Listing 6.** is a short program that will allow you to reverse specified rectangles on the Hi-Res screen (perhaps for no reason other than to see how fast it can do the entire screen).

LISTING 1

```
SOURCE FILE: FILL
----- NEXT OBJECT FILE NAME IS FILL.OBJ0
0300:                1          ORG  $300
DEBE:                2 CHKCOM   EQU  $DEBE
F6F0:                3 HCOLOR   EQU  $F6F0
E74C:                4 COMBYTE  EQU  $E74C
F6B9:                5 HFNS     EQU  $F6B9
F457:                6 HPLOT    EQU  $F457
F53A:                7 HLIN     EQU  $F53A
0300:                8 ;
0300:                9 ; GET COLOR
0300:               10 ;
0300:20 4C E7       11          JSR  COMBYTE
0303:20 F0 F6       12          JSR  HCOLOR
0306:20 BE DE       13          JSR  CHKCOM
0309:               14 ;
0309:               15 ; GET TOP-LEFT HAND CORNER OF THE BOX
0309:               16 ;
0309:20 B9 F6       17          JSR  HFNS
030C:8E 57 03       18          STX  FIRSTL
030F:8C 58 03       19          STY  FIRSTH
0312:8D 59 03       20          STA  TOP
0315:20 BE DE       21          JSR  CHKCOM
0318:               22 ;
0318:               23 ; GET BOTTOM-RIGHT HAND CORNER OF THE BOX
0318:               24 ;
0318:20 B9 F6       25          JSR  HFNS
031B:8E 5A 03       26          STX  SECONDL
031E:8C 5B 03       27          STY  SECONDH
0321:8D 5C 03       28          STA  BTM
0324:               29 ;
0324:               30 ; GET NUMBER OF SPACES BETWEEN LINES
0324:               31 ;
0324:20 4C E7       32          JSR  COMBYTE
0327:E8             33          INX
0328:8E 5D 03       34          STX  SPACES
032B:               35 ;
032B:               36 ; FILL IN THE BOX WITH HORIZONTAL LINES
032B:               37 ;
032B:AE 57 03       38 START    LDX  FIRSTL
032E:AC 58 03       39          LDY  FIRSTH
0331:AD 59 03       40          LDA  TOP
0334:20 57 F4       41          JSR  HPLOT
0337:AE 5B 03       42          LDX  SECONDH
033A:AD 5A 03       43          LDA  SECONDL
033D:AC 59 03       44          LDY  TOP
0340:20 3A F5       45          JSR  HLIN
0343:AD 59 03       46          LDA  TOP
0346:18             47          CLC
0347:6D 5D 03       48          ADC  SPACES
034A:8D 59 03       49          STA  TOP
034D:AD 5C 03       50          LDA  BTM
0350:38             51          SEC
0351:ED 59 03       52          SBC  TOP
0354:B0 D5          53          BCS  START
0356:60             54          RTS
0357:               55 FIRSTL   DS   $1
0358:               56 FIRSTH   DS   $1
0359:               57 TOP      DS   $1
035A:               58 SECONDL  DS   $1
035B:               59 SECONDH  DS   $1
035C:               60 BTM      DS   $1
035D:               61 SPACES   DS   $1

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

LISTING 2

```
]LIST

10  HOME
20  PRINT  CHR$ (4)"BLOAD FILL.OBJ0"
30 FILL = 768
40  HGR
50  VTAB 21
60  INPUT "COLOR & NUMBER OF SPACES ? ";COLR,SP
70  INPUT "LEFT COLUMN & TOP ROW ? ";LC,TR
80  INPUT "RIGHT COLUMN & BOTTOM ROW ? ";RC,BR
90  CALL FILL,COLR,LC,TR,RC,BR,SP
100  HCOLOR= 3: HPLOT LC,TR: HPLOT  TO RC,TR TO RC,BR TO
     LC,BR TO LC,TR: HPLOT LC + 1,TR TO LC + 1,BR: HPLOT
     RC + 1,TR TO RC + 1,BR
110  GOTO 50
```

LISTING 6

```
]LIST

10  HOME
20  PRINT  CHR$ (4)"BLOAD REVERSE HI RES.OBJ0"
30 REVERSE = 864
40  HGR
50  VTAB 21
70  INPUT "LEFT COLUMN & TOP ROW ? ";LC,TR
80  INPUT "RIGHT COLUMN & BOTTOM ROW ? ";RC,BR
90  CALL REVERSE,LC,TR,RC,BR

100  GOTO 50
```

LISTING 3

```
]LIST

0  HOME :FILL = 768
0  PRINT  CHR$ (4)"BLOAD FILL.OBJ0"
0  HGR2 :N = 142: GOSUB 1000
0 M =  INT ( RND (1) * N) + 1:L = L(M):T = T(M):L(M) =
     L(N):T(M) = T(N)
5 Z =  INT ( RND (1) * 8):COLR = CO(Z)
0  CALL FILL,COLR,L,T,L + 5,T + 5,0
5 X =  PEEK ( - 16336) +  PEEK ( - 16336)
0 N = N - 1: IF N ) 0 THEN 50
5  GET A$: HOME : TEXT : CALL 62450: END
0  END
000  DIM L(142),T(142)
005  FOR I = 0 TO 7:CO(I) = I: NEXT :CO(0) = 2:CO(4) =
     6:CO(7) = 3
010  FOR I = 1 TO 142: READ L(I),T(I): NEXT
015  CALL FILL,3,24,0,29,32,0: CALL FILL,3,30,12,45,1
     7,0: CALL FILL,3,40,18,45,32,0: CALL FILL,3,30,27
     ,40,32,0
017  CALL FILL,3,52,12,57,32,0: CALL FILL,3,58,27,67,
     32,0: CALL FILL,3,68,12,73,48,0: CALL FILL,3,42,4
     3,67,48,0
019  RETURN
020  DATA 32,72,24,72,16,72,8,72,0,72,0,80,0,88,0,96,
     0,104,0,112,8,112,16,112,24,112,32,112,32,104,32,
     96,24,96,16,96
030  DATA 48,96,48,88,48,80,48,72,48,64,56,56,64,56,7
     2,56,80,64,80,72,80,80,80,88,80,96,56,80,64,80,72
     ,80
040  DATA 96,88,96,80,96,72,96,64,96,56,96,48,104,48,
     112,48,120,48,128,56,128,64,120,72,112,72,104,72,
     128,80,128,88
050  DATA 144,80,144,72,144,64,144,56,144,48,144,48,1
     52,48,160,48,168,48,176,48,176,56,168,64,160,64,1
     52,64,160,72,174,80
060  DATA 204,88,204,80,204,72,204,64,196,56,188,48,2
     12,56,220,48
070  DATA 32,136,32,144,32,152,32,160,32,168,32,176,6
     4,136,56,144,48,152,48,160,48,168,56,168,64,176
080  DATA 88,144,88,152,88,160,88,168,88,176,88,184
090  DATA  112,144,120,144,128,144,136,144,144,144,14
     4,152,136,160,128,168,120,176,112,184,120,184,128
     ,184,136,184,144,184
000  DATA 168,136,168,144,168,152,168,160,168,168,168
     ,176
010  DATA 192,168,192,160,192,152,192,144,192,136,200
     ,128,208,128,216,128,224,136,224,144,224,152,224,
     160,224,168,208,152,208,152,216,152
020  DATA 240,120,240,128,248,136,240,144,240,152,240
     ,160,272,128,264,128,256,136,248,144,256,144,264,
     152,272,160
```

LISTING 4

```
OURCE FILE: REVERSE HI RES
---- NEXT OBJECT FILE NAME IS REVERSE HI RES.OBJ0
360:                1          ORG  $360
06:                 2 LEFTCOL  EQU  $06
07:                 3 TOPLINE  EQU  $07
08:                 4 RIGHTCOL EQU  $08
09:                 5 BTMLINE  EQU  $09
26:                 6 HBASL    EQU  $26
411:                7 HPOSN    EQU  $F411
4C:                 8 COMBYTE  EQU  $E74C
360:                9 ;
360:               10 ; GET PARAMETERS AND STORE IN ZERO PAGE
360:               11 ;
360:A0 00          12          LDY  #$00
0362:98            13 INIT     TYA
0363:48            14          PHA
0364:20 4C E7      15          JSR  COMBYTE
0367:68            16          PLA
0368:A8            17          TAY
0369:96 06         18          STX  LEFTCOL,Y
036B:C8            19          INY
036C:C0 04         20          CPY  #$04
036E:D0 F2         21          BNE  INIT
0370:              22 ;
0370:              23 ; START AT THE BOTTOM LINE
0370:              24 ;
0370:A6 09         25          LDX  BTMLINE
0372:8A            26 START    TXA
0373:48            27          PHA
0374:              28 ;
0374:              29 ; FIND THE BASE ADDRESS OF THE HI RES LINE
0374:              30 ;
0374:20 11 F4      31          JSR  HPOSN
0377:68            32          PLA
0378:AA            33          TAX
0379:              34 ;
0379:              35 ; REVERSE THAT LINE
0379:              36 ;
0379:A4 08         37          LDY  RIGHTCOL
037B:B1 26         38 TOP      LDA  (HBASL),Y
037D:49 FF         39          EOR  #$FF
037F:91 26         40          STA  (HBASL),Y
0381:88            41          DEY
0382:C4 06         42          CPY  LEFTCOL
0384:10 F5         43          BPL  TOP
0386:E4 07         44          CPX  TOPLINE
0388:F0 04         45          BEQ  END
038A:              46 ;
038A:              47 ; WORK YOUR WAY UP TO THE TOP LINE
038A:              48 ;
038A:CA            49          DEX
038B:18            50          CLC
038C:90 E4         51          BCC  START
038E:60            52 END      RTS

***.SUCCESSFUL ASSEMBLY: NO ERRORS
```