DRAWING HI-RES CHARACTERS

The Graphics Workshop takes a short time-out from its exploration of block shapes this month to consider a perennial problem for graphics programmers: displaying text on the Hi-Res screen. Using the combination Applesoft and machine language program presented, you will be able to add Hi-Res text horizontally, vertically and even diagonally! You can even expand the Shape Table to include your own special characters.

by Robert A. Devine
1415 W. 19th Street
El Dorado, AK

INTRODUCTION

If you do a lot of Hi-Res graphics programming, like I do, you've probably run into the problem of wanting to print text, or some special messages on the graphics screen. But sometimes placing all of your communications in those four bottom lines of text doesn't make for a professional looking program. If you're displaying graphics on Hi-Res page 2, then you're totally out of luck, so some better answer is needed.

To solve the problem, I bought a Mountain Computer ROMPLUS board with a keyboard filter. This hardware is great, but let's face it — if you're writing programs which may run on someone else's Apple (I write a lot for *Nibble*), your program can't depend on these aids. The solution is to build your own software-driven general purpose graphics aids.

SPECIAL CHARACTERS

In this article, we'll look at a Shape Table that makes a large variety of special graphics characters available for use on either Hi-Res screen. These special characters are:

1. The numbers 0-9.
2. The alphabet characters A-Z.
3. The special characters **?.,!'+-=$** and a blank.
4. An apple.
5. The symbols representing the four suits of playing cards: the diamond, heart, spade, and club. These characters are included in the Shape Table listing which I've called **GRAPHICS CHAR A$9300/L768.** All of the shapes in this table are **vector** shapes.

"PRINTING" WITH SPECIAL CHARACTERS

So you may say, BIG DEAL!!! If you've been using your Apple for very long, you've probably already run into several listings of keyboard Shape Tables that can be used to simulate Hi-Res text. What's different about this one? Nothing, really!

We're going to concentrate here on some techniques that you can use, with this table or others, to make "printing" with these tables almost as easy as if you were using normal text. We'll look at an Applesoft program listing which demonstrates several uses of these characters on the Hi-Res screen, as well as a subroutine which shows how you can easily access any of these characters for printing on the screen.

With our routines you will be able to . . .

PRINT IN THE NORMAL FASHION, OR

YOU CAN PRINT THIS WAY,

OR ALONG THE SIDE.

OR EVEN DIAGONALLY!!

OR YOU CAN EVEN PRINT UP-SIDE DOWN.

In the demonstration, you'll see examples of how to DRAW the characters using normal HPLOT coordinates, as well as a way to easily simulate the VTAB and HTAB statements used in regular **text** printing.

TYPING IN THE PROGRAM

To enter the program, first type in the character Shape Table from **Listing 1** (see the Welcome to New Nibble Readers if you don't know how) and save it to disk with the following command:

BSAVE GRAPHICS CHAR A$9300/L768, A$9300,L768

Then key in the demonstration program from **Listing 2** and save it to disk with the command **SAVE DRAW TEXT DEMO**. RUN the program to see how it works, and then examine it to see the programming techniques used.

HOW THE TABLE IS ORGANIZED

First let's look at how the table is organized, how the shapes were created, and how you can add your own shapes to the table.

The first byte of every vector Shape Table indicates the number of shapes in the table; in this case we've indicated that there are 53 shapes. In reality there are only 50, as one of the characters in the table is a blank. Character 11 (the blank) has no shape; therefore, no vectors are needed. So pointer 11 simply points to the last two bytes in the table, $95FE and $95FF, both of which should contain 00. If you add any shapes to the table, you can't use these two bytes, nor can you use any bytes past them, because DOS begins at the very next byte, $9600.

The reason that 53 shapes are listed is that there are 29 bytes ($95E1-$95FD) that are available for you to add up to two additional shapes of your own, while retaining the balance of the table. To add them, simply put the shapes in the referenced bytes. The pointer for shape 52 is already set, but you'll need to set the pointer for shape 53 in bytes $936A and $936B.

Bytes $9300-$936B are the 53 shape pointers.

Bytes $936C-$93DB are shapes 1-10 (the numbers 0-9).

Bytes $93DC-$94F4 are shapes 12-37 (the alphabet A-Z).

Bytes $94F5-$9534 are shapes 38-46 (the characters **?.,!'+-=$**).

Bytes $9535-$9569 are shape 47 (an apple).

Bytes $9570-$95E0 are shapes 48-51 (diamond, heart, spade, and club).

Bytes $95E1-$95FD are available for shapes 52 and 53.

Bytes $95FE-$95FF are shape 11 (a blank).

HOW THE SHAPES ARE CREATED

All of the normal keyboard characters are drawn on a 5x7 grid and we have used this same grid to draw our Hi-Res shapes. If you wish to add other characters that will be compatible, you should be careful to begin your new shape in the top left-hand corner of the 5x7 box so the characters will line up properly. The playing card symbols and the apple are larger, and each of these begins in the center of the shape.

THE DEMONSTRATION PROGRAM

In the demonstration program you will find four separate demos, each showing different ways of using the character set. We won't take any time going through the demos themselves. Your best bet here is to enter the program and run all the demos. You can then go into each clearly-marked demo section to see how the various types of printing were accomplished.

WHAT MAKES IT ALL WORK

The heart of the program is contained in **lines 30 and 40** as well as in **lines 460-500**. These are the parts of the program that you should understand in order to apply these techniques for your own use.

Let's look at **lines 460-500** first, as they get everything set up for the program.

Line 460 BLOADs the Shape Table and sets HIMEM. You should note that HIMEM is set not only to protect the Shape Table, but also to protect an ASCII Value Table and

short machine code shape translator that we'll also be POKEing into memory.

Line 470 sets the Shape Table pointers and initializes SCALE, ROTation, and HCOLOR.

Line 480 enters a special ASCII Value Table into memory that will be the key to all translating of keyboard characters into the proper shape numbers. First we set A$ to a string that includes all the legal characters, placed in the same order as they appear in the table. Then we step through the string and POKE the proper ASCII code for each keyboard character into the ASCII Value Table.

Line 490 reads and POKEs a short machine code translator into memory. The following describes how that routine disassembles and what it does:

92BF- A2 2E	LDX #46	Point to last keyboard character in the table.
92C1- BD D0 92	LDA $92D0,X	Get Xth ASCII code from the ASCII Value Table.
92C4- C5 19	CMP $19	Compare to ASCII code in $19 (dec. 25).
92C6- F0 05	BEQ $92CD	If it matches, GOTO $92CD.
92C8- CA	DEX	Point to next code in ASCII Value Table.
92C9- D0 F6	BNE $92C1	If not at end of table, GOTO $92C1.
92CB- A2 0B	LDX #11	No match found — set for shape 11 (blank).
92CD- 86 19	STX $19	Store X-Register value in $19 (dec. 25) shape number.
92CF- 60	RTS	Shape number selected — < RETURN >

In order to use the routine, you must POKE the ASCII code for the character that you want to print into location 25 ($19), then CALL 37567 which is the translator routine. This routine uses the X-Register to step through the ASCII Value Table, looking for a code that matches what you put in memory location 25. When it finds a match, it replaces the value in location 25 with the contents of the X-Register, which is the proper shape number for that character. If it goes completely through the ASCII Value Table without finding a match, it returns with the value 11 (a blank) in location 25, thus eliminating the possibility of your program crashing due to illegal input.

The proper syntax for use with the translator is:

POKE 25, ASC(character you wish to print)
CALL 37567
DRAW PEEK(25) at X,Y

This same translation method could easily be applied to any other Shape Table of keyboard characters that you might have.

SIMULATING THE HTAB AND VTAB STATEMENTS

One big problem of trying to print text on the Hi-Res screen is that while the Hi-Res screen is 280 dots wide by 192 dots high (or 160 dots high on page 1), the normal text screen is only 40 characters wide by 24 lines high. In order to make it easy to work with, we'll simply assume that we're dealing with the normal text screen which has legal HTABs of 0-39 and legal VTABs of 0-23.

We will use the variable HT to specify the HTAB where we want to begin printing our string, and VT to specify the VTAB where our string will appear.

The proper syntax for printing a string (in its normal horizontal position) is:

HT=HTAB (0-39)
VT=VTAB (0-23) or **0-19 on Hi-Res page 1**
A$="STRING YOU WISH TO PRINT"
GOSUB 30

Line 30 will automatically translate your HT and VT into the proper Hi-Res coordinates for the DRAWing routines and move your "cursor" rightward one position for each new character in the str ng. The string will be pulled apart, character-by-character, translated into the proper shape numbers, and printed on the screen.

The print routine at **line 30** is intended for use with normal left-right printing. When playing our games with printing upside-down and backwards, or up and down on the screen, you will need to break apart the string to be printed somewhere else in your program and send the characters to be printed to the print routine, at **line 30**, character-by-character.

A CLOSING NOTE

If you don't have much experience working with Shape Tables, or you've had times when you wanted to print text on the Hi-Res screen, it would be worth your time and effort to enter and run the demo listing. You may later find some of the techniques quite useful.

LISTING 1 — DRAW TEXT DEMO

```
1  REM   ************************
2  REM   *   DRAW TEXT  DEMO    *
3  REM   * BY ROBERT R. DEVINE *
4  REM   * COPYRIGHT (C)  1984 *
5  REM   * BY MICROSPARC, INC. *
6  REM   * LINCOLN, MA.  01773 *
7  REM   ************************
10  GOSUB 460: GOTO 60
20  REM  LINES 30-40 SET HTAB & VTAB, TRANSLATE STRING
     ELEMENTS TO SHAPE #S, TEST FOR ILLEGAL CHARACTER
    S, AND PRINT THE STRING
30 HT = HT * 7:VT = VT * 8: FOR X = 1 TO  LEN (A$): POKE
    25, ASC ( MID$ (A$,X,1)): CALL 37567: REM  PULL S
    TRING APART, CALL TRANSLATOR TO GET SHAPE #
40  DRAW  PEEK (25) AT (HT - 7) + (X * 7),VT: NEXT : RETURN

50  REM  **** DEMO #1 ****
60  HOME : HGR2 :A$ = "HI-RES CHARACTER DRAWING":VT =
    2:HT = 8: GOSUB 30
70 A$ = "WITH THIS ROUTINE YOU CAN 'PRINT' ANY":VT = 4
    :HT = 0: GOSUB 30
80 A$ = "OF THE FOLLOWING CHARACTERS.....":VT = 5: GOSUB
    30
90 A$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ":VT = 7: GOSUB 30
100 A$ = "0123456789 ?.,!'+-=$":VT = 8: GOSUB 30
110 A$ = "OR THESE HANDY SHAPES.....":VT = 10: GOSUB 3
    0
120 VT = VT + 15:Y = 0: FOR X = 47 TO 51:Y = Y + 1: DRAW
    X AT Y * 15,VT: NEXT X
130 A$ = "THE 'PRINT' ROUTINE SIMULATES THE":VT = 13: GOSUB
    30
140 A$ = "NORMAL VTAB AND HTAB FUNCTION.":VT = 14: GOSUB
    30
150 A$ = "IF YOU TRY TO ENTER AN ILLEGAL CHARACTER":VT
     = 16: GOSUB 30
160 A$ = "A BLANK WILL APPEAR IN IT'S PLACE.":VT = 17:
    GOSUB 30
170 A$ = "TOUCH ANY KEY TO CONTINUE-":VT = 20: GOSUB 3
    0: GET A$
180  REM  **** DEMO #2 ****
190  HOME : HGR : VTAB 22: HTAB 5: PRINT "HOW ABOUT A
    HANDY SALES CHART ?"
200  HPLOT 40,0 TO 40,140 TO 275,140: FOR X = 20 TO 11
    6 STEP 24: FOR Y = 40 TO 270 STEP 10: HPLOT Y,X TO
    Y + 1,X: NEXT Y,X
210  DRAW 47 AT 98,10:A$ = "COMPUTER SALES":VT = 1:HT =
    16: GOSUB 30:T =  - 1: FOR Z = 100 TO 20 STEP  -
    20:HT = 2:T = T + 3:VT = T:A$ =  STR$ (Z): GOSUB
    30: NEXT Z
220 B$ = "SALES IN THOUSANDS":T = 19: ROT= 48: FOR Z =
    1 TO 18:A$ =  MID$ (B$,Z,1):T = T - 1:VT = T:HT =
    0: GOSUB 30: NEXT Z: ROT= 0
230 A$ = "J  F  M  A  M  J  J  A  S  O  N  D":VT = 18:
    HT = 6: GOSUB 30
240  HPLOT 42,100 TO 63,50 TO 84,40 TO 105,55 TO 126,6
    0 TO 147,20 TO 168,25 TO 189,50 TO 210,70 TO 231,
    30 TO 252,35 TO 273,25
250  HOME : VTAB 22: HTAB 6: PRINT "TOUCH ANY KEY TO C
    ONTINUE )";: GET A$: PRINT
260  REM  **** DEMO #3 ****
270  HOME : HGR2 :A$ = "ANYBODY FOR A GAME OF CARDS ?"
    :VT = 0:HT = 5: GOSUB 30
280 B$ = "A500K4852510Q4958485A4854500":B = 1
290  FOR Z = 0 TO 240 STEP 40: GOSUB 440: HCOLOR= 0
300  POKE 25, ASC ( MID$ (B$,B,1)): CALL 37567
310  DRAW  PEEK (25) AT Z + 5,53: DRAW  VAL ( MID$ (B$
    ,B + 1,2)) AT Z + 12,56: ROT= 32: DRAW  PEEK (25)
     AT Z + 32,107: DRAW  VAL ( MID$ (B$,B + 1,2)) AT
    Z + 25,104: ROT= 0
320 B = B + 4: NEXT Z: HCOLOR= 3: HPLOT 0,150 TO 0,190
     TO 279,190 TO 279,150 TO 0,150
330 A$ = "HUMAN'S SCORE = " +  STR$ ( INT ( RND (1) *
    500)):VT = 20:HT = 10: GOSUB 30
340  DRAW 47 AT 80,178:A$ = "'S SCORE = " +  STR$ ( INT
    ( RND (1) * 500)):VT = 22:HT = 13: GOSUB 30
350 A$ = "TOUCH ANY KEY TO CONTINUE":VT = 17:HT = 7: GOSUB
    30: GET A$: PRINT
360  REM  **** DEMO #4 ****
370  HOME : HGR2 :A$ = "NOW YOU CAN PRINT ANY LETTERS,
    ":VT = 0:HT = 0: GOSUB 30
380 B$ = "ANYWHERE YOU WANT THEM":T = 0: ROT= 16: FOR
    Z = 1 TO 22:A$ =  MID$ (B$,Z,1):T = T + 1:VT = T:
    HT = 31: GOSUB 30: NEXT Z: ROT= 32
390 B$ = "ON THE HI-RES SCREEN.":T = 29: FOR Z = 1 TO
    21:A$ =  MID$ (B$,Z,1):VT = 23:HT = T: GOSUB 30:T
     = T - 1: NEXT Z
400 A$ = "END OF DEMO !!": ROT= 54:X = 40:Y = 160:Z =
    1
410  POKE 25, ASC ( MID$ (A$,Z,1)): CALL 37567
420  DRAW  PEEK (25) AT X,Y:Z = Z + 1: ON (Z = 15) GOTO
    430:X = X + 10:Y = Y - 10: GOTO 410
430  GET K$: TEXT : HOME : END
440  HCOLOR= 3: FOR A = Z TO Z + 37: HPLOT A,50 TO A,1
    10: NEXT A: HCOLOR=  VAL ( MID$ (B$,B + 3,1)): FOR
    A = Z + 4 TO Z + 32: HPLOT A,65 TO A,95: NEXT A: RETURN
450  REM   THIS ROUTINE READS THE SHAPE TABLE,AND SETS
     POINTERS
460  PRINT  CHR$ (4)"BLOAD GRAPHICS CHAR A$9300/L768":
     HIMEM: 37567
470  POKE 232,0: POKE 233,147: SCALE= 1: ROT= 0: HCOLOR=
    3
480 A$ = "0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ?.,!'+-
    =$":Y = 37585: FOR X = 1 TO  LEN (A$): POKE Y, ASC
    ( MID$ (A$,X,1)):Y = Y + 1: NEXT X: REM  POKE ASC
     VALUE TABLE INTO MEMORY
490  FOR X = 37567 TO 37583: READ Y: POKE X,Y: NEXT : RETURN
500  DATA  162,46,189,208,146,197,25,240,5,202,208,246
    ,162,11,134,25,96
```

```
KEY PERFECT 4.0
RUN ON
DRAW TEXT DEMO
========================================
     CODE        LINE# - LINE#
     ------      -------------
     AEAD          1 -   30
     BFDC         40 -  130
     FCE8        140 -  230
     F18F        240 -  330
     D3D4        340 -  430
     CD15        440 -  500
TOTAL PROGRAM CHECK IS : 0A91


CHECK CODE 3.0

ON: DRAW TEXT DEMO
TYPE: A

LENGTH: 09DB
CHECKSUM: 20
```

LISTING 2 — GRAPHICS CHAR A$9300/L768

```
9300- 35 00 6C 00 7A 00 84 00
9308- 8E 00 99 00 A5 00 B1 00
9310- BC 00 C4 00 D0 00 FE 02
9318- DC 00 E8 00 F4 00 FE 00
9320- 09 01 15 01 20 01 2B 01
9328- 37 01 41 01 4B 01 57 01
9330- 5E 01 6B 01 78 01 83 01
9338- 8C 01 98 01 A4 01 B0 01
9340- B7 01 C0 01 CA 01 D6 01
9348- E2 01 EB 01 F5 01 FD 01
9350- 02 02 08 02 0D 02 12 02
9358- 1A 02 1F 02 27 02 35 02
9360- 70 02 8A 02 A9 02 C3 02
9368- E1 02 00 00 32 36 76 2D
9370- 0C 24 24 1C 3F 4E BA 17
9378- 06 00 12 2C 2C 36 36 6E
9380- 1A 3F 3F 00 62 2D 15 BE
9388- 17 BF 2E 2D 2D 00 62 2D
9390- 15 FE 2A 15 F6 3F 1C 04
9398- 00 49 36 AE 37 26 1C 3F
93A0- 27 21 21 21 00 2D 2D DE
93A8- 1B 36 2D AD F6 3F 1C 04
93B0- 00 09 F5 BB 36 76 2D 0C
93B8- E4 3F 07 00 66 2D 35 17
93C0- 17 36 36 00 29 AD B6 F6
93C8- 3F 1C 64 2D 3F 1C 24 00
93D0- 32 0E 2D 9E 13 65 0C 24
93D8- E4 3F 07 00 29 AD 36 36
93E0- FE 1B 24 24 AC 2A 35 00
93E8- 2D AD B6 F6 3F 27 24 24
93F0- 95 2D 06 00 29 AD 96 32
93F8- 3B E7 24 24 04 00 2D AD
9400- 36 36 1E 3F 27 24 24 04
9408- 00 2D 2D 96 92 3F 3F 24
9410- 24 AC 2A 2D 00 2D 2D DF
9418- 1B 36 2D F5 1B 36 05 00
9420- 29 AD 97 35 F6 3F 1C 24
9428- 24 04 00 36 36 36 4D 21
9430- 24 3F 67 09 24 04 00 2D
9438- 2D DE 36 36 9F 2D 2D 04
9440- 00 49 2D 1E 36 36 1E 3F
9448- 1C 04 00 36 36 36 4D E1
9450- 1C 1C 0C 0C 0C 04 00 36
9458- 36 36 2D 2D 05 00 36 36
9460- 36 4D 21 24 24 BC 1E 1C
9468- 56 06 00 36 36 36 4D 21
9470- 24 24 FC 93 15 15 06 00
9478- 29 AD 36 36 1E 3F 1C 24
9480- 24 04 00 2D AD F6 3F 27
9488- B4 32 36 00 29 AD 36 B6
9490- 1C 1C 16 E7 24 24 04 00
9498- 36 36 36 4D E1 1C 7C 65
94A0- E4 3F 07 00 29 AD DF 33
94A8- 0E 2D 15 F6 3F 1C 04 00
94B0- 2D 2D DE 36 36 36 00 36
94B8- 36 76 2D 0C 24 24 24 00
94C0- 36 36 15 15 0C 0C 24 24
94C8- 04 00 36 6E FE 36 0C 0D
94D0- 35 24 24 24 04 00 2E AA
94D8- 17 17 6E 09 E4 1C 0C 0C
94E0- 24 00 36 15 96 21 64 0C
94E8- 24 04 00 2D 2D BE 17 17
94F0- 17 2E 2D 2D 00 62 2D 15
94F8- BE 17 16 05 00 92 52 35
9500- 3F 00 92 52 31 1E 07 00
9508- 09 36 B6 32 00 29 36 27
9510- 04 00 12 65 AC 11 3F 36
9518- 06 00 92 2D 2D 05 00 12
9520- 2D 2D 16 3F 3F 07 00 09
9528- 8D 3F BF 0D 15 3F 0E 0D
9530- 1E 3F 77 29 00 92 36 2D
9538- 2C 3F 3F 3E 3C 3E 38 28
9540- 2D 6D 2D 2D 38 3F 3F 3F
9548- 3F 3F 2C 2D 2D 2D 2D 2D
9550- 3C 3F 3F 3F 3F 3F 2C 2D
9558- 2D 2D 2D 2D 3C 3F 3F 3F
9560- 3F 3F 0C 2D 2D 2D 2D E5
9568- FF FF 67 09 25 2D 2D 00
9570- 25 25 25 25 15 3E 3E 3E
9578- 3E 0E 25 25 25 25 15 3E
9580- 3E 3E 3E 0E 2C 2C 2C 2C
9588- 05 00 24 25 0C 36 3E 36
9590- 0E 24 25 24 15 36 3E 36
9598- 0E 24 35 0C 3C 24 24 0C
95A0- 36 36 35 0C 24 1C 36 36
95A8- 00 76 24 64 36 36 0C 24
95B0- 24 0C 36 36 36 BE 2D 1C
95B8- 64 24 24 15 36 36 25 24
95C0- 15 36 00 76 24 2C 36 2E
95C8- 20 24 3C 2C 2C 36 36 36
95D0- 36 17 2D 1C 24 25 24 24
95D8- 2C 32 16 36 2E 24 AC 36
95E0- 00 00 00 00 00 00 00 00
95E8- 00 00 00 00 00 00 00 00
95F0- 00 00 00 00 00 00 00 00
95F8- 00 00 00 00 00 00 00 00
```

```
KEY PERFECT 4.0
RUN ON
GRAPHICS CHAR A$9300/L768
========================================
     CODE        ADDR# - ADDR#
     ------      -------------
     2AD4        9300 - 934F
     234F        9350 - 939F
     23A0        93A0 - 93EF
     246B        93F0 - 943F
     280E        9440 - 948F
     2A56        9490 - 94DF
     29CD        94E0 - 952F
     25DA        9530 - 957F
     24D0        9580 - 95CF
     164A        95D0 - 95FF
TOTAL PROGRAM CHECK IS : 0300


CHECK CODE 3.0

ON: GRAPHICS CHAR A$9300/L768
TYPE: B

LENGTH: 0300
CHECKSUM: A5
```