

Memory Addressing the Apple Graphics Screen

E.S. deSa*, and E.J. D'Sa*

1. Introduction

When the Apple II is set in the High Resolution graphics mode, it can display 53,750 dots in a matrix defined by 280 dots wide and 192 dots high. As is commonly known, the primary page of the picture buffer (invoked by the Applesoft HGR command) is a 8192 byte area of memory covering the address range 8192 to 16393. Similarly, the secondary page (invoked by the HGR2 command) extends over the address range 16394 to 24575.

Frequently, in writing graphics software on the Apple, it is useful to know relationship between a dot's (x,y) coordinates to its memory address in the graphics area. A glance at the memory map of the high-res screen of the Apple shows that the organisation of addresses do not follow a well-ordered sequential scheme over the full memory page. However, by dividing the screen matrix into three separate memory blocks, the memory address of any dot belonging to a byte can be related to its (x,y) coordinates on the Apple screen matrix by the following relations.

For the HGR page:

$$M1 = 8192 + Qx + Qy * 128 + 1024 * 8 * Ry \quad (1)$$

valid over the coordinate range $(0,0) \leq (x,y) \leq (279,63)$

$$M2 = 8232 + Qx + (Qy - 8) * 128 + 1024 * 8 * Ry \quad (2)$$

Valid over the coordinate range $(0,64) \leq (x,y) \leq (279,127)$

$$M3 = 8272 + 6B Qx + (Qy - 16) * 128 + 1024 * 8 * Ry \quad (3)$$

Valid over the coordinate range $(0,128) \leq (x,y) \leq (129,159)$ and excluding the text window.

The addresses M1, M2 & M3 correspond to the three memory blocks comprising the HGR page, and

Qx = Quotient of $(x/7)$

Qy = Quotient of $(y/8)$

Ry = Remainder of $(y/8)$

In like manner, the HGR2 page is defined by the equations:

$$N1 = 16384 + Qx + (Qy) * 128 + Ry * 1024 * 8 \quad (4)$$

$$N2 = 16424 + Qx + (Qy - 8) * 128 + Ry * 1024 * 8 \quad (5)$$

$$N3 = 16464 + Qx + (Qy - 16) * 128 + Ry * 1024 * 8 \quad (6)$$

The quantities Qx , Qy & Ry and coordinate ranges are as given for the HGR page, except that for block N3 (equation 6) the coordinates (x,y) range from (0,128) to (279,192).

Figure 1 shows the high-resolution graphics area partitioned into three separate memory blocks. There are eight box rows in each block, and each box row has eight "line rows". A line row is 40 bytes wide with sequential addresses running along any line row.

Further examination of the screen matrix shows that for a given block and a fixed column:

- * Corresponding line rows between neighbouring box rows are separated by 128 bytes.
- * Neighbouring line rows of a given box row are separated by 1024 bytes.

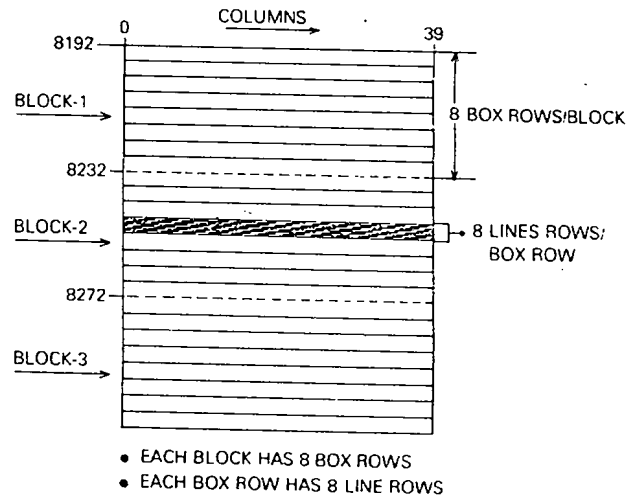


Fig.1 HGR map of the Apple high-resolution Screen

Instrumentation and Computer Division,
National Institute of Oceanography,
Dona Paula, Goa 403 004.

With these facts in mind, it is easy to deduce the equations listed above. The variable Q_x signifies the column number in the range 0 to 39; $Q_y + 128$ selects the box row number; $R_y * 1024 * 8$ selects the line row within the selected box row. In this way the address of any block data byte in graphics memory can be computed.

A simple nest of FOR — NEXT LOOPS will enable a user to examine or modify address contents in any selected memory block, e.g.

```

10 FOR I = 0 TO 7          : Box row counter
20 FOR J = 0 to 7         : Line row counter
30 FOR K = 0 to 39       : Column counter
40 MI = 8192 + 128 * I + 1024 * J + K
50 POKE (MI), 255
60 X = PEEK (MI)
70 NEXT K: NEXT J: NEXT I
    
```

2. Use of Memory Equations in Horizontal Scrolling

As a further example of the use of the above equations, an Applesoft Basic program was written to implement horizontal scrolling of a single wave form. The program list out is shown in Appendix I.

The program plots a sine wave in the memory block I of the Apple HGR page for coordinates in the range (0,0) to (279,63). Whilst plotting the (x,y) points of the sine wave, the address GADD of a data point, column number Q_x and data byte LU are determined and stored in the matrix arrays M, C and D.

By pressing the right cursor key → the right scroll routine is executed at line 700. In order to do right scrolling, the waveform needs to be translated into a fixed number of bytes, D. Each successive press of the cursor key adds an integral number of D bytes to the addresses of data points in the original waveform over successively smaller plotting sections of the curve. In our case, each key press removes eight data addresses from the total number of data addresses.

The left scroll routine executes at line 500 after a key press of the left cursor key ←. It is similar to right scrolling except that the shift of amount D bytes is subtracted from the addresses of original data points.

3. Summary

The graphics address equations derived here provide a useful tool when writing graphics software. An example of its use in horizontal scrolling was implemented without having to write a lengthy machine language routine. As a simple extension, a BASIC program could be written to implement vertical scrolling of a graphics picture using these equations.

APPENDIX I: A Sample Program

```

5 REM: HSCROLL
10 REM: SINE WAVE
20 REM: -----
30 DIM M(300)
32 DIM D(300)
33 DIM C(300)
35 HGR: HCOLOR = 3
40 K = 0:L = 0:XM = 720:R = 0
45 REM: -----
50 FOR XR = 0 TO XM STEP 10
60 YR = .5 * SIN ((3.1417 * XR/180) + 1.5708) + .5
70 XG = INT (279 * XR/XM): YG = INT (63 - YR * 63)
80 H PLOT XG, YG
90 QY = INT (YG/8): QX = INT (XG/7)
100 RY = (YG/8) - INT (YG/8)
110 GADD = 8192 + Qx + 128 * QY + 1024 * 8 * RY
120 LU = PEEK (GADD)
130 M(K) = GADD: D(K) = LU:C(K) = QX
135 PRINT GADD, LU, QX
140 K = K + 1
150 NEXT XR
155 REM: -----
160 NUM = K
170 PRINT "NO.ELEMS = ":NUM
180 REM : -----
190 INVERSE
200 PRINT "→SCROLL RIGHT:: ← SCROLL LEFT"
205 NORMAL
206 REM : -----
210 GET C$
220 IF ASC (C$) = 8 THEN GOSUB 500
230 IF ASC (C$) = 21 THEN GOSUB 700
240 IF ASC (C$) = 13 THEN END
250 GOTO 210
260 REM : -----
500 HGR : HCOLOR =3
510 L = L + 1
515 OFF = C(8 * L)
520 FOR K = 8 * L TO NUM
530 POKE (M(K) - OFF), D(K)
540 NEXT K
550 RETURN
560 REM : -----
700 HGR : HCOLOR = 3
710 R = R + 1
715 OFF = 39 - C(NUM - 8 * R)
720 FOR K = 0 TO (NUM - 8 * R)
730 POKE (M(K) + OFF), D(K)
740 NEXT K
750 RETURN
760 REM : -----
    
```