Apple II Technical Notes

Developer Technical Support

Apple IIe
#3: Double High-Resolution Graphics

Revised by: Matt Deatherage, Glenn A. Baxter & Cameron Birse
November 1988
Written by: Peter Baum
September 1983

This Technical Note is a tutorial on double high-resolution (hi-res) graphics, a feature available on 128K Apple IIe, IIc, and IIGS computers.

Introduction

This Note was originally written in the early days of double high-resolution graphics. At that time, there was no Apple IIc or IIGS, therefore, some of the things originally said may seem a little strange today, five years later.

For example, this Note talks a fair amount about being sure that you have a Revision B Apple IIe with the jumper installed. All Apple IIe computers shipped since about mid-1983 have a Revision B motherboard, so this is not that big a concern anymore; furthermore, nearly every IIe out there has the aforementioned jumper already installed (it is not even an option on some third-party 80-column cards for the IIe).

Also, the IIc and IIGS are functionally equivalent (for the purposes of this article) to a Revision B IIe with the properly-jumpered 80-column card installed, and most of the references made to the Apple IIe apply equally to the IIc and IIGS. We have tried to update most of the references to avoid confusion.

Considering the myriad of programming utilities, games, graphics programs, and other software that now uses double high-resolution graphics, it is probable that this Note will not be as vital as it once was. If you are writing in AppleSoft BASIC, you will probably find it easier to purchase

a commercial double hi-res BASIC utility package to add double hi-res commands to AppleSoft, rather than writing your own routines. Similarly, those who want double hi-res art will find a double hi-res art application much easier than trying to draw it from the monitor or machine language.

However, if you have the insatiable curiosity about these things that Apple II owners and developers so often are blessed (cursed?) with, this Note will show you how double high-resolution works, as well as giving a few type-along examples in the monitor to get your feet wet.

This article describes the double high-resolution display mode which is available in the Apple IIc, IIGS, and the Apple IIe (with an extended 80-column card). Double hi-res graphics provides twice the horizontal resolution and more colors than the standard high-resolution mode. On a monochrome monitor, double hi-res displays 560 horizontal by 192 vertical pixels, while on a color monitor, it allows the use of 16 colors.

Double High-Resolution on the Apple II Series

What is It?

The double high-resolution display mode that is available for the Apple IIe provides twice the horizontal resolution of the standard high-resolution mode. On a standard black-and-white video monitor, standard hi-res displays 280 columns and 192 rows of picture elements (pixels); the double hi-res mode displays 560 x 192 pixels. On a color monitor, the standard hi-res mode displays up to 140 columns of colors, each color being selected from the group of six colors available, with certain limitations. Double hi-res displays 140 columns of color, for which all 16 of the low-resolution colors are available.

	Black/White	Color
Standard Hi-Res	280 x 192 pixels	140 columns 6 colors
Double Hi-Res	560 x 192 pixels	140 columns 16 colors

Table 1–Comparison of Standard and Double Hi-Res Graphics

How Do I Install It?

Installation of the double hi-res mode on your Apple IIe depends on the following three conditions, discussed in detail below:

1. Presence of a Revision B motherboard
2. Installation of an extended 80-column text card with jumper
3. A video monitor with a bandwidth of at least 14 MHz

First, your Apple IIe must have a Revision B (Rev-B) motherboard. To find out whether your computer's motherboard is a Rev-B, check the part number on the edge of the board nearest the back panel, above the slots. If the board is a Rev-B, the part number will be 820-0064-B. (Double hi-res does not work on systems containing a Rev-A motherboard.) If your computer's motherboard is not a Rev-B, and if you want to obtain one, contact your local Apple dealer.

The second condition for installing double hi-res on your IIe is that it must have an extended 80-column text card installed. This card must be installed with a jumper connecting the two Molex-type pins on the board.

Warning: If your IIe has a Rev-A motherboard, do **not** use an extended 80-column

card with the jumper connection mentioned above; the system will not work at all if you do.

The last requirement for operation in double hi-res mode is that your video monitor must have a bandwidth of at least 14 MHz. This bandwidth is necessary because a television set that requires a modulator will not reproduce some characters or graphic elements clearly, due to the high speed at which the computer sends out dots in this mode. Because most of the video monitors having a bandwidth of up to 14 MHz are black-and-white, the working examples in this article do not apply to color monitors. If you have a video monitor, please use it—instead of a television set—to display the following examples. The AppleColor composite monitors will work just fine.

Your Turn to be Creative (Volunteers, Anyone?)

The tutorial that occupies the rest of this Note assumes you are working at your Apple II as you read. The second part of the lesson demonstrates the double hi-res mode; therefore, before embarking on the second part, you should install a jumpered extended 80-column card in your Rev-B Apple IIe (or use any Apple IIc or IIGS).

Hands-On Practice with Standard Hi-Res

The Apple II hi-res graphics display is bit-mapped. In other words, each dot on the screen corresponds to a bit in the computer's memory. For a real-life example of bit-mapping, perform the following procedure, according to the instructions given below. (The symbol <cr> indicates a carriage return.)

1. Boot the system.
2. Engage the Caps Lock key, and type `HGR<cr>`. (This instruction should clear the top of the screen.)
3. Type `CALL -151 <cr>`. (The system is now in the monitor mode, and the prompt should appear as an asterisk (*).)
4. Type `2100:1 <cr>`. One single dot should appear in the upper left-hand corner of the screen.

Congratulations! You have just plotted your first hi-res pixel. (Not an astonishing feat, but you have to start somewhere.)

With a black-and-white monitor, the bits in memory have a simple correspondence with the dots (pixels) on the screen. A dot of light appears if the corresponding bit is set (has a value of 1), but remains invisible if the bit is off (has a value of zero). (The dot appears white on a black-and-white monitor, and green on a green-screen monitor, such as Apple's Monitor /// or Monitor II. For simplicity, we shall refer to an invisible dot as a black dot or pixel.) Two visible dots located next to each other appear as a single wide dot, and many adjacent dots appear as a line. To obtain a display of another dot and a line, follow the steps listed below:

1. Type `2080:40 <cr>`. A dot should appear above and to the right of the dot you produced in the last exercise.
2. Type `2180:7F <cr>`. A small horizontal line should appear below the first dot you produced.

From Bits and Bytes to Pixels

The seven low-order bits in each display byte control seven adjacent dots in a row. A group of

40 consecutive bytes in memory controls a row of 280 dots (7 dots per byte, multiplied by 40 bytes). In the screen display, the least-significant bit of each byte appears as the leftmost pixel in a group of 7 pixels. The second least-significant bit corresponds to the pixel directly to the right of the pixel previously displayed, and so on. To watch this procedure in action, follow the steps listed below. The dots will appear in the middle of your screen.

1. Type `2028:1 <cr>`.
2. Type `2828:2 <cr>`.
3. Type `3028:4 <cr>`.

The three bits you specified in this exercise correspond to three pixels that are displayed one after another, from left to right.

The most-significant bit in each byte does not correspond to a pixel. Instead, this bit is used to shift the positions of the other seven bits in the byte. For a demonstration of this feature, follow the steps listed below:

1. Type `2050:8 <cr>`.
2. Type `2850:8 <cr>`.
3. Type `3050:8 <cr>`.

You will notice that the dots align themselves vertically. Now do the following:

4. Type `2450:88 <cr>`.

The new dot (that is, the one that corresponds to the bit you just specified) does not line up with the dots you displayed earlier. Instead, it appears to be shifted one "half-dot" to the right.

5. To demonstrate that this dot really is a new dot, and not just the old dot shifted by one dot position, type `2050:18 <cr>`, `2850:18 <cr>`.

You will notice that the dot mentioned under step 4 (the dot that was not aligned with the other seven dots) is straddled by the dots above and below it. (The use of magnifying lenses is permitted.)

Shifting the pixel one half-dot, by setting the high, most-significant bit is most often used for color displays. When the high bit of a byte is set to generate this shifted dot (which is also called the half-dot shift), then all the dots for that byte will be shifted one half dot. The half-dot shift does not exist in the double hi-res mode.

The Figure 1 shows the memory map for the standard hi-res graphics mode.

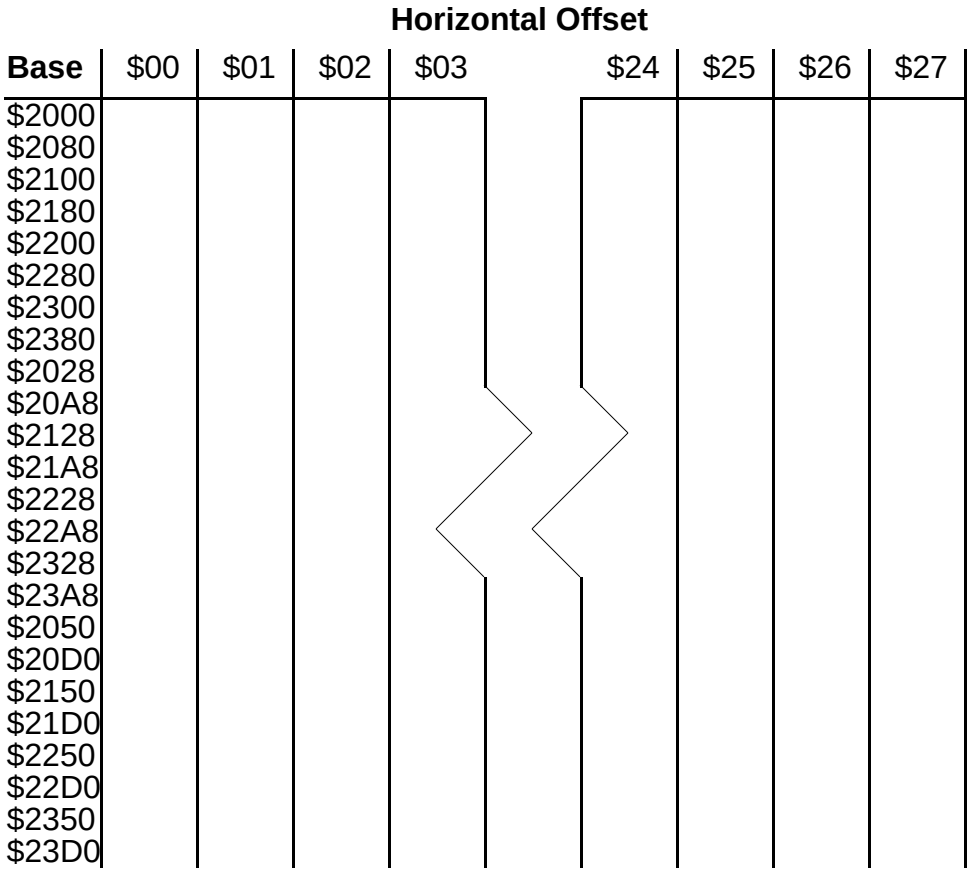

Figure 1–Standard Hi-Res Memory Map

Figure 2 shows the box subdivisions for the memory map in Figure 1.

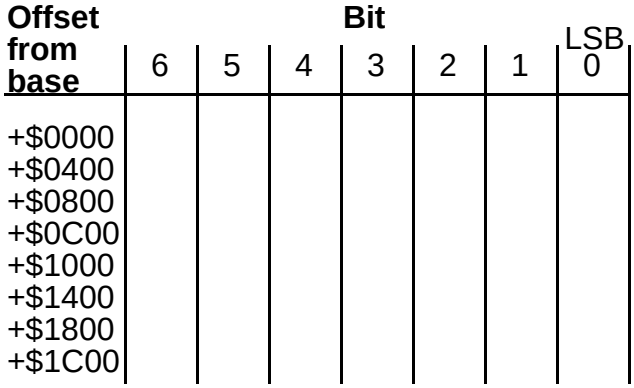

Figure 2–Box Subdivisions of the Standard Memory Map

For example, the first memory address of each screen line for the first few lines is as follows:

$2000, $2400, $2800, $2C00, $3000, $3400, $3800, $3C00, $2080, $2480, etc.

Each of the 24 boxes contains 8 screen lines for a total of 192 vertical lines per screen. Each of

the 40 boxes per line contains 7 pixels for a total of 280 pixels horizontally across each line.

The Intricacies of Double Hi-Res

Because the double high-resolution graphics mode provides twice the horizontal dot density as standard hi-res graphics does, double hi-res requires twice as much memory as does standard hi-res. If you spent many hours committing the standard hi-res memory map to memory, don't despair; double hi-res still uses the hi-res graphics page (but only to represent half the picture, so to speak). In the double hi-res mode, the hi-res graphics page is compressed to fit into half of the display. The other half of the display is stored in memory (called the auxiliary (aux) memory) on the extended 80-column card. (This article refers to the standard hi-res graphics page, which resides in main memory, as the motherboard (main) memory.)

The auxiliary memory uses the same addresses used by the standard hi-res graphics page (page 1, $2000 through $3FFF). The hi-res graphics page stored in auxiliary memory is known as hi-res page 1X. The graphics pages in auxiliary memory are bank-switched memory, which you can switch in by activating some of the soft switches. (Adventurous readers may want to skip ahead to Using the Auxiliary Memory, which appears later in this Note.)

The memory mapping for the hi-res graphics display is analogous to the technique used for the 80-column display. The double hi-res display interleaves bytes from the two different memory pages (auxiliary and motherboard). Seven bits from a byte in the auxiliary memory bank are displayed first, followed by seven bits from the corresponding byte on the motherboard. The bits are shifted out the same way as in standard hi-res (least-significant bit first). In double hi-res, the most significant bit of each byte is ignored; thus, no half-dot shift can occur. (This feature is important, as you will see when we examine double hi-res in color.)

The memory map for double hi-res appears in Figure 3.

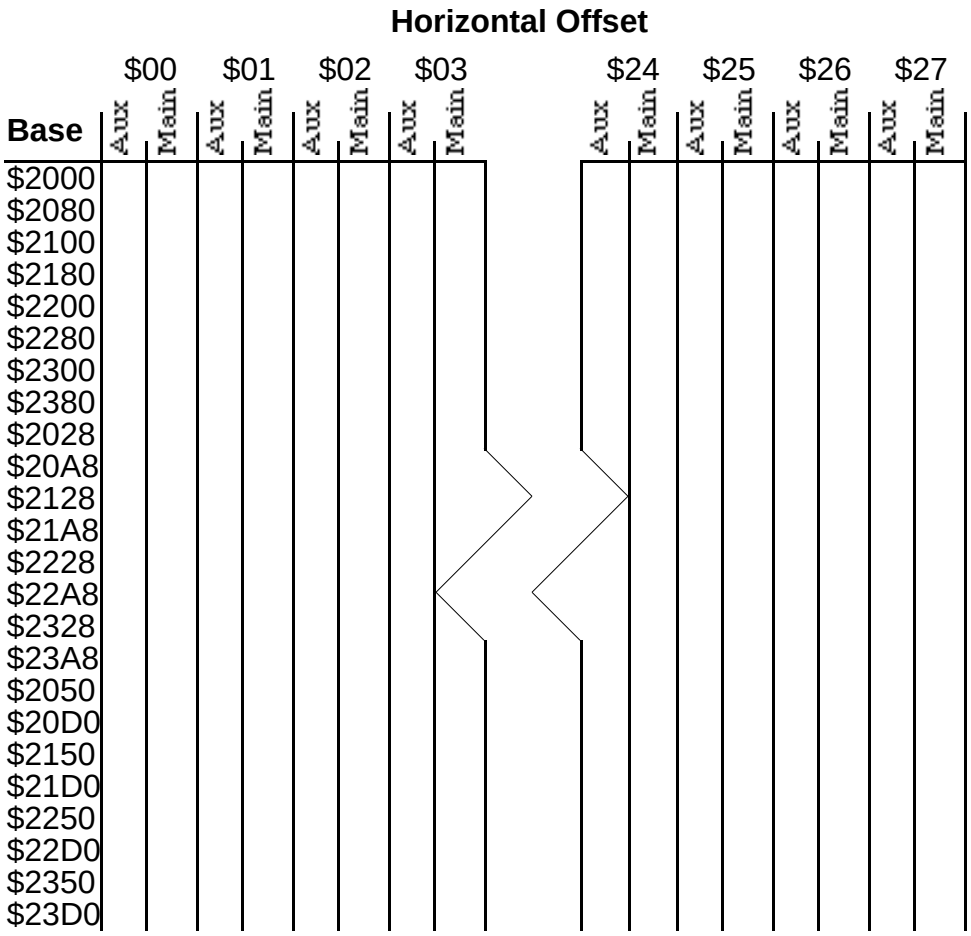

Figure 3–Double Hi-Res Memory Map

Each box is subdivided exactly the same way it is in the standard hi-res mode.

Obtaining a Double-Hi-Res Display

To display the double hi-res mode, set the following soft switches:

	In the monitor Read	In AppleSoft PEEK
HI-RES	$C057	49239
GR	$C050	49232
AN3	$C05E	49246
MIXED	$C053	49235

	In the monitor Write	In AppleSoft POKE
80COL	$C00D	49165,0

Annunciator 3 (AN3) must be turned off to get into double hi-res mode. You turn it off by reading location 49246 ($C05E). Note that whenever you press Control-Reset, AN3 is turned on; therefore, each time you press Control-Reset, you must turn AN3 off again.

If you are using MIXED mode, then the bottom four lines on the screen will display text. If you have not turned on the 80-column card, then every second character in the bottom four lines of text will be a random character. (The reason is that although the hardware displays 80 columns of characters, the firmware only updates the 40-column screen, which consists of the characters in the odd-numbered columns. The characters in even-numbered columns then consist of random characters taken from text page 1X in the auxiliary memory.)

To remove the even characters from the bottom four lines on the screen, type PR#3<CR> from AppleSoft (type 3^P in the monitor). This procedure clears the memory locations on page 1X.

Using the Auxiliary Memory

The auxiliary memory consists of several different sections, which you can select by using the soft switches listed below. A pair of memory locations is dedicated to each switch. (One location turns the switch on; the other turns it off.) You activate a switch by writing to the appropriate memory location. The write instruction itself is what activates the switch; therefore, it does not matter what data you write to the memory location. The soft switches are as follows:

		In the monitor Write	In AppleSoft POKE
80STORE	off	$C000	49152,0
	on	$C001	49153,0
RAMRD	off	$C002	49154,0
	on	$C003	49155,0
RAMWRT	off	$C004	49156,0
	on	$C005	49157,0
PAGE2	off	$C054	49236,0
	on	$C055	49237,0
HIRES	off	$C056	49238,0
	on	$C057	49239,0

A routine called AUXMOVE ($C311), located in the 80-column firmware, is also very handy, as we will see below.

Accessing memory on the auxiliary card with the soft switches has the following characteristics. Memory maps, which help clarify the descriptions, are in Figures 4, 5, and 6.

1. To activate the PAGE2 and HIRES switches, you need only read (PEEK) from the corresponding memory locations (instead of writing to them, as you do for the other three switches).

2. The PAGE2 switch normally selects the display page, in either graphics or text mode, from either page 1 or page 2 of the motherboard memory. However, it does so only when the 80STORE switch is off.
3. If the 80STORE switch is on, then the function of the PAGE2 switch changes. When 80STORE is on, then PAGE2 switches in the text page, locations $400-7FF, from auxiliary memory (text page 1X), instead of switching the display screen to the alternate video page (page 2 on the motherboard). When 80STORE is on, the PAGE2 switch determines which memory bank (auxiliary or motherboard) is used during any access to addresses $400 through 7FF. When the 80STORE switch is on, it has priority over all other switches.

4. If the 80STORE switch is on, then the PAGE2 switch only switches in the graphics page 1X from the auxiliary memory if the HIRES switch is also on. (Note that this circumstance is slightly different from that described in item 3.) When 80STORE is on, and if the HIRES switch is also on, then the PAGE2 switch selects the memory bank (auxiliary or motherboard) for accesses to a memory location within the range $2000 through 3FFF. If the HIRES switch is off, then any access to an memory location within the range $2000 through 3FFF uses the motherboard memory, regardless of the state of the PAGE2 switch.
5. If the 80STORE switch is off, and if the RAMRD and RAMWRT switches are on, then any reading from or writing to address space $200-$BFFF gains access to the auxiliary memory. If only one of the switches, RAMRD, for example, is set, then only the appropriate operation (in this case a read) will be performed on the auxiliary memory. If only RAMWRT is set, then all write operations access the auxiliary memory. When the 80STORE switch is on, it has higher priority than the RAMRD and RAMWRT switches.

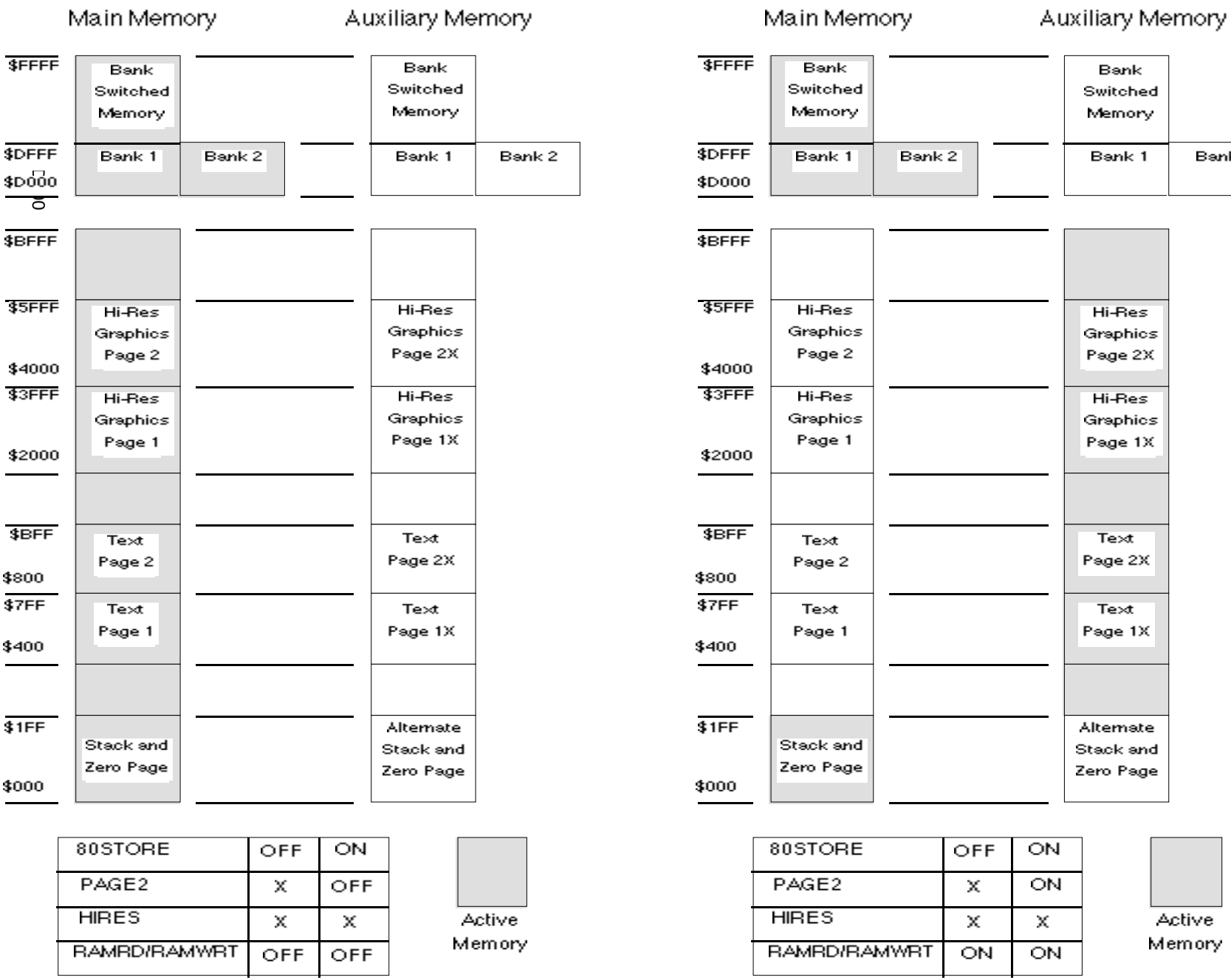

Figure 4–Memory Map One

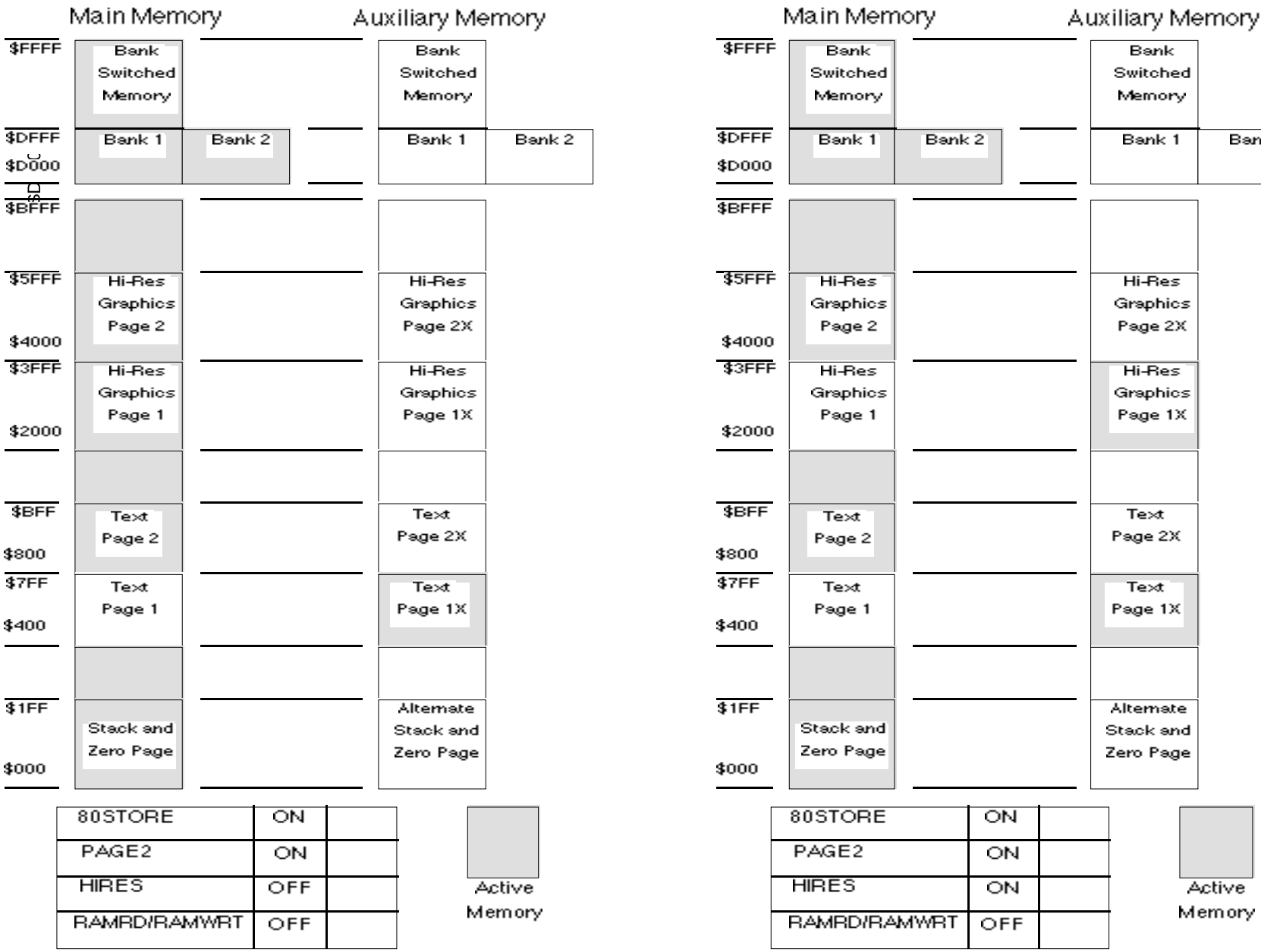

Figure 5–Memory Map Two

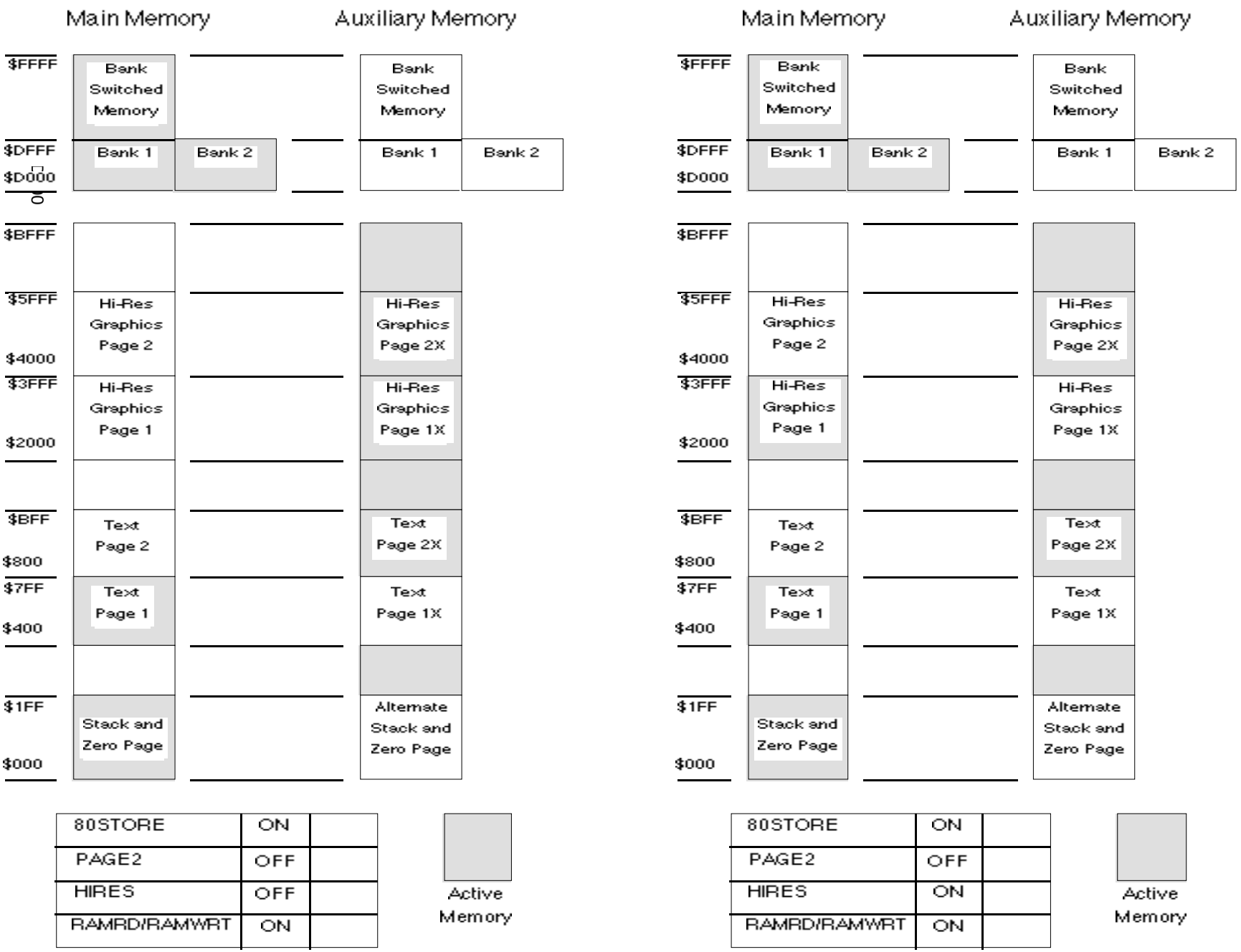

Figure 6–Memory Map Three

Shortcuts: Writing to Auxiliary Memory from the Keyboard

Press Control-Reset, then type `CALL -151 <cr>` (to enter the monitor). Now type the following hexadecimal addresses to turn on the double hi-res mode:

C057	(for hi-res)
C050	(for graphics)
C053	(for mixed mode)
C05E	Turns off `AN3` for double hi-res
C00D:0	Turns on the `80COL` switch

This procedure usually causes the display of a random dot pattern at the top of the screen, while the bottom four lines on the screen contain text. To clear the screen, follow the steps listed below:

1. Type `3D0G <cr>` to return to BASIC.
2. Type `HGR <cr>` to clear half of the screen. (The characters you type will probably appear in alternating columns. This is not a cause for alarm; as noted above, the firmware simply thinks you are working with a 40-column display.) Remember that hi-res graphics commands do not know about the half of the screen stored on page 1X in the auxiliary memory. Therefore, only page 1 (that is, the first half) of the graphics page on the motherboard is cleared. As a result, in the the screen display, only alternate 7-bit columns appear cleared.

   On the other hand, if all of the screen columns were cleared after the `HGR` command, then chances are good that you are not in double hi-res mode. If your screen was cleared then to determine which mode you are in, type the following instructions:

```
CALL -151
2000:FF
2001<2000.2027M
```

   If a solid line appears across the top of the screen, you are not in double hi-res mode. (The line that appears should be a dashed or intermittent line: - - - - - across the screen.) If you are not in double hi-res mode, then make sure that you do have a Rev. B motherboard, and that the two Molex-type pins on the extended 80-column card are shorted together with the jumper block. Then re-enter the instructions listed above.

If you are staring at a half-cleared screen, you can clear the non-blank columns by writing zeros to addresses $2000 through $3FFF on graphics page 1X of auxiliary memory. To do so, simply turn on the 80STORE switch, turn on the PAGE2 switch, then write to locations $2000, $2001, $2002, and so on up through $3FFF. However, this procedure will not work if you try it from the monitor. The reason is that each time you invoke a monitor routine, the routine sets the PAGE2 switch back to page 1 so that it can display the most recent command that you entered. When you try to write to $2000, etc. on the auxiliary card, instead it will write to the motherboard memory.

Another way to obtain the desired result is to use the monitor's USER command, which forces a jump to memory location $3F8. You can place a JMP instruction starting at this memory location, so the program will jump to a routine that writes into hi-res page 1X. Fortunately, the system already contains such a routine: AUXMOVE.

Using AUXMOVE

You use the AUXMOVE routine to move data blocks between main and auxiliary memory, but the task still remains of setting up the routine so that it knows which data to write, and where to write it. To use this routine, some byte pairs in the zero page must be setup with the data block addresses, and the carry bit must be fixed to indicate the direction of the move. You may not be surprised to learn that the byte pairs in the zero page used by AUXMOVE are also the scratch-pad registers used by the monitor during instruction execution. The result is that while you type the addresses for the monitor's move command, those addresses are being stored in the byte pairs used by AUXMOVE. Thereafter, you can call the AUXMOVE command directly, using the USER (Control-Y) command.

In practice, then, enter the following instructions:

```
C00A:0              (turns on the 80-column ROM, which contains the
                    AUXMOVE routine)
C000:0              (reason explained below)
3F8: 4C 11 C3       (the jump to AUXMOVE)
2000<2000.3FFF ^Y   (where ^Y indicates that you should type Control-Y)
```

The syntax for this USER (Control-Y) command is:

```
{AUXdest}<{MBstart}.{MBend}^Y
```

The command copies the values in the range MBstart to MBend in the motherboard memory into the auxiliary memory beginning at AUXdest. This command is analogous to the MOVE command.

You can use this procedure to transfer any block of data from the motherboard memory to hi-res page 1X. Working directly from the keyboard, you can use a data block transferred this way to fill in any part of a double hi-res screen image. The image to be stored in hi-res page 1X (i.e., the image that will be displayed in the even-numbered columns of the double hi-res picture) must first be stored in the motherboard memory. You can then use the Control-Y command to transfer the image to hi-res page 1X.

The AUXMOVE routine uses the RAMRD and RAMWRT switches to transfer the data blocks. Because the 80STORE switch overrides the RAMRD and RAMWRT switches, the 80STORE switch must be turned off—otherwise it would keep the transfer from occurring properly (hence the write to $C000 above).

If the 80STORE and HIRES switches are on and PAGE2 is off, when you execute AUXMOVE, any access to an address located within the range from $2000 to $3FFF inclusive would use the motherboard memory, regardless of how RAMRD and RAMWRT are set. Entering the command C000:0 <cr> turns off 80STORE, thus letting the RAMRD and RAMWRT switches control the memory banking.

The Control-Y trick described above only works for transferring data blocks from the main (motherboard) memory to auxiliary memory (because the monitor always enters the AUXMOVE routine with the carry bit set). To move data blocks from the auxiliary memory to the main memory, you must enter AUXMOVE with the carry bit clear. You can use the following routine to transfer data blocks in either direction:

```
301:AD 0 3          (loads the contents of address $300 into the accumulator)
304:2A              (rotates the most-significant bit into the carry flag)
305:4C 11 C3        (jump to $C311 (AUXMOVE))
3F8:4C 1 3          (sets the Control-Y command to jump to address $301)
```

Before using this routine, you must modify memory location $300, depending on the direction in which you want to transfer the data blocks. If the transfer is from the auxiliary memory to the motherboard, you must clear location $300 to zero. If the transfer is from the motherboard to the auxiliary memory, you must set location $300 to $FF.

Two Double Hi-Res Pages

So far, we have only discussed using graphics pages 1 and 1X to display double hi-res pictures. But—analogous to the standard hi-res pages 1 and 2—two double hi-res pages exist: pages 1 and 1X, at locations $2000 through 3FFF, and pages 2 and 2X, at locations $4000 through 5FFF. The only trick in displaying the second double hi-res page is that you must turn off the 80STORE switch. If the 80STORE switch is on, then only the first page (1 and 1X) is displayed. Go ahead and try it:

```
C000:0          to turn off the 80STORE switch
C055            to turn on the PAGE2 switch
```

The screen will fill with another display of random bits. Clear the screen using the instructions listed above (in the Using AUXMOVE section). However, this time, use addresses $4000 through 5FFF instead. (Don't be alarmed by the fact that the figures you are typing are not displayed on the screen. They are being "displayed" on text page 1.)

```
4000:0
4001<4000.5FFFM
4000<4000.5FFF ^Y
```

You will be delighted to learn that you can also use this trick to display two 80-column text screens. The only problem here is that the 80-column firmware continually turns on the 80STORE switch, which prevents the display of the second 80-column screen. However, if you write your own 80-column display driver, then you can use both of the 80-column screens.

Color Madness

It should come as no surprise that color-display techniques in double hires are different from color-display techniques in standard hi-res. This difference is because the half-dot shift does

not exist in double hi-res mode.

Instead of going into a dissertation on how a television set decodes and displays a color signal, I'll simply explain how to generate color in double hi-res mode. In the following examples, the term color monitor refers to either an NTSC monitor or a color television set. Both work; however, the displays will be much harder to see on the color television. The generation of color in double hi-res demands sacrifices. A 560 x 192 dot display is not possible in color. Instead, the horizontal resolution decreases by a factor of four (140 dots across the screen). Just as with a black-and-white monitor, a simple correspondence exists between memory and the pixels on the screen. The difference is that four bits are required to determine each color pixel. These four bits represent 16

different combinations: one for each of the colors available in double hi-res. (These are the same colors that are available in the low-resolution mode.)

Let's start by exploring the pattern that must be stored in memory to draw a single colored line across the screen. Use a color demonstration program (such as COLOR.TEST from older DOS 3.3 System Master disks) to adjust the colors displayed by your monitor. After you have adjusted the colors, exit from the color demonstration program.

The instructions that appear below are divided into groups separated by blank lines. Because it is very difficult (and, on a television set, almost impossible) to read the characters you are typing as they appear on the screen, you will probably make typing errors. If the instructions appear not to work, then start again from the beginning of a group of instructions.

```
CALL -151          (to get into the monitor routine/program)
C050               (This set of instructions puts the computer
CO57               into double hi-res model.
C05E
C00D:0
2000:0             (This set of instructions clears first one half
2001<2000.3FFFM    of the screen, and then the other half of
3F8: 4C 11 C3      the screen.)
2000<2000.3FFF^Y

2100:11 4          (Two red dots appear on top left of screen)
2102<2100.2126M    (A dashed red line appears across screen)

2150:8 22          (Two green dots appear near bottom left)
2152<2150.2175M    (Dashed green line appears across screen)

2100<2150.2177^Y   (Fills in the red line)
```

In contrast to conditions in standard hi-res, no half-dot shift occurs, and the most-significant bit of each byte is not used.

As noted above, four bits determine a color. You can paint a one-color line across the screen simply by repeating a four-bit pattern across the screen, but it is much easier to write a whole byte rather than just change four bits at a time. Since only seven bits of each byte are displayed (as noted earlier in our discussion of black-and-white double hi-res) and the pattern is four bits wide, it repeats itself every 28 bits or four bytes. Use the instructions listed below to draw a line of any color across the screen by repeating a four byte pattern for the color as shown in Table 2.

```
2200: main1 main2
```
(Colored dots appear at the left edge)

```
2202<2200.2226M
```
(A dashed, colored line appears)

```
2250: aux1 aux2
2250<2250.2276M
```

```
2200<2250.2276^Y
```
(Fills in line, using the selected color)

Color	aux1	main1	aux2	main2	Repeated Binary Pattern
Black	00	00	00	00	0000
Magenta	08	11	22	44	0001
Brown	44	08	11	22	0010
Orange	4C	19	33	66	0011
Dark Green	22	44	08	11	0100
Grey1	2A	55	2A	55	0101
Green	66	4C	19	33	0110
Yellow	6E	5D	3B	77	0111
Dark Blue	11	22	44	08	1000
Violet	19	33	66	4C	1001
Grey2	55	2A	55	2A	1010
Pink	5D	3B	77	6E	1011
Medium Blue		33	66	4C	19 1100
Light Blue	3B	77	6E	5D	1101
Aqua	77	6E	5D	3B	1110
White	7F	7F	7F	7F	1111

Table 2–The Sixteen Colors

In Table 2, the heading aux1 indicates the first, fifth, ninth, thirteenth, etc. byte of each line (i.e., every fourth byte, starting with the first byte). The heading main1 indicates the second, sixth, tenth, fourteenth, etc. byte of each line (i.e., every fourth byte, starting with the second byte). The aux2 and main2 headings indicate every fourth byte, starting with the third and fourth bytes of each line, respectively. Aux1 and aux2 are always stored in auxiliary memory, while main1 and main2 are always stored in the motherboard memory.

As you will infer from Table 2, the absolute position of a byte also determines the color displayed. If you write an 8 into the first byte at the far left side of the screen (i.e., in the aux1 column), then a red dot is displayed. But if you write an 8 into the third byte at the left side of the screen (the aux2 column), then a dark green dot is displayed. Remember, the color monitor decides which color to display based on the relative position of the bits on each line (i.e., on how far the bits are from the left edge of the screen).

So far, so good. But suppose you want to display more than one color on a single line. It's easy: just change the four-bit pattern that is stored in memory. For example, if you want the left half of the line to be red, and the right half to be purple, then store the red pattern (8, 11, 22, 44) in the first 40 bytes of the line, then store the purple pattern (19, 33, 66, 4C) in the second 40 bytes of the line. Table 2 is a useful reference tool for switching from one color to another, provided you make the change on a byte boundary. In other words, you must start a new color

at the same point in the pattern at which the old color ended. For example, if the old color stops after you write a byte from the main1 column, then you should start the new color by storing the next byte in memory with a byte from the aux2 column. This procedure is illustrated below:

```
2028:11 44 11 44 11 44 11 77 5D 77 5D 77 5D      (creates a dashed line
2128: 8 22 8 22 8 22 8 22 6E 3B 6E 3B 6E     that is red then yellow)

2028<2128.2134^Y                                 (fills in the rest of the colors)
```

Switching Colors in Mid-Byte

If you want a line to change color in the middle of a byte, you will have to recalculate the column, based on the information in Table 2. Suppose you want to divide the screen into three vertical sections, each a different color. The leftmost third of the screen ends in the middle of the 27th character from the left edge–that is, in an aux2 column of the color table. (Dividing 27 by 4 gives a remainder of 3, which indicates the third column, or aux2.) Your pattern should change from the first color to the second color after the 5th bit of the 27th byte. You can change the color in the middle of a byte by selecting the appropriate bytes from the aux2 column of Table 2 and concatenating two bits for the second color with five bits for the first color.

However, because the bits from each byte are shifted out in order from least significant to most significant, the two most significant bits (in this case bits 5 and 6, because bit 7 is unused) for the second color are concatenated with the five least significant bits for the first color. For instance, if you want the color to change from orange (the first color) to green (the second color), then you must append the two most significant bits (5 and 6) of green to the five least significant bits (0–4) of orange. In Table 2, the aux2 column byte for green is 19, and the two most significant bits are both clear. The aux2 column byte for orange is 33, and the five least significant bits are equal to 10011. The new byte calculated from appending green (00) to orange (10011) yields 13 (0010011). Therefore, the first 26 bytes of the line come from the table values for orange; the 27th byte is 13, and the next 26 bytes come from the table values for green.

```
2300: 19 66                          (puts an orange line on the screen)
2302<2300.2310M
2350: 4C 33
2352<2350.2360M
2300<2350.2360^Y

230D: 33 4C 33 4C 33 4C 33 4C (puts a green line next to it)
235D: 13 66 19 66 19 66 19 66 (note the first byte)
230D<235D.2363^Y
```

There you have it: a basic explanation of how double hi-res works—except for one or two anomalies. The first anomaly is that NTSC monitors have a limited display range. The second anomaly shows one of the features of double hi-res versus a limitation of standard hi-res.

An NTSC color monitor decides what color to display based on its view of four bit windows in each line, starting from the left edge of the screen. The monitor looks at the first four bits, determines which color is called for, then shifts one bit to the right and determines the color for this new four-bit window. But remember, the color depends not only on the pattern, but also the position of the pattern. To compensate for relative position from the left edge of the screen, the

monitor keeps track of where on each line each of these windows start. (For those of you of the technical persuasion, this is done through the use of the color burst signal, which is a 3.58 MHz. clock).

Try this example:

```
2000:0                  Clears the screen
2001<2000.3FFFM
2000<2000.3FFF^Y
```

```
2001:66                 Draws an orange box in the upper left
2401:66
2801:66
2C01:66
3001:66

2050:33                 Draws a blue box below and to the right
3402<2050.2050^Y        of the orange box
3802<2050.2050^Y
3C02<2050.2050^Y
```

Notice that if the blue box was drawn at the top of the screen, next to the orange box, they would overlap. Yet, the boxes were drawn on two different columns, orange on main2 and blue on aux1. This can be explained by the previous paragraph, and the sliding windows. The monitor will detect the pattern for orange slightly after the main2 column, while the pattern for blue shows up before column aux1.

The orange pattern is as follows:

```
0000000 | 0110011 |0000000      look at four-bit windows and you will see
  aux2 |    main2 |    aux1 an orange pattern overlaps on both sides
```

If a pattern is repeated on a line, this overlap does not cause a problem, since the same color just overlaps itself. But watch what happens when a new pattern is started next to a different pattern:

```
3002<2050.2050^Y            Puts a blue pattern next to the orange one
2C02<2050.2050^Y
2802<2050.2050^Y
```

Where the blue overlaps the orange, you will see a white dot. This effect is because one of the four-bit windows the monitor sees is all 1s. If two colors are placed right next to each other, the monitor will sometimes display a third color, or fringe, at the boundary. This fringe effect is especially noticeable when there are a lot of narrow columns of different colors next to each other. (Next time you run COLOR TEST take a look at the boundaries between the colors).

The orange and blue patter is as follows:

```
0000000 |0110011 | 11001100    note the four 1s in a row at the boundary
  aux2 |  main2 |      aux1     between orange and blue
```

Conclusion

Now you have the tools and the rules to the double hi-res mode. As you can see, double hi-res has more color with higher resolution than standard hi-res. You can even develop games that do fancy animation or scroll orange objects across green backgrounds. You can develop word processing programs which use different fonts or proportional character sets in black and white. Have fun playing with his new mode.

Further Reference

- *Apple IIe Technical Reference Manual*
- *Apple IIc Technical Reference Manual, Second Edition*
- *Apple IIGS Hardware Reference*