PIXEL PLOTTING

We begin a series of articles exploring graphics applications using 6502 and Z80 machine code on the more popular home computers. Here, we discuss the use of 6502 Assembly language to access the Commodore 64's high resolution screen.

The various steps and procedures involved in Commodore high resolution graphics have been thoroughly explored on page 254: we must first switch the Video Interface Chip (VIC) to high resolution mode, and change the character set base address pointer; the block of eight Kbytes starting at location 8192 that will hold the screen memory map must be cleared; and the normal screen memory map (locations 1024 to 2033 — $0400 to $07E7), which is now to be used to hold screen colour information, must be initialised. This last task can be complicated in multi-colour displays, but is straightforward here since we shall have only one background colour on the screen.

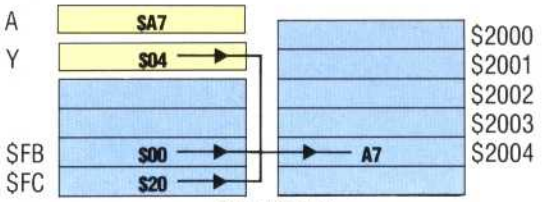
The program will allow switching into and out of high resolution mode, and then perform all the necessary calculations to plot a point on the high resolution screen. So the first part of our program will concern itself with the two important tasks that have to be carried out before the high resolution screen can be used: the colour information has to be put into each of the normal screen locations, and the eight Kbyte bit map has to be cleared.

To allow the same routine to be called when entering or leaving high resolution mode, a flag called HRSFLG will be used. In addition, we may not always wish to clear the bit map on entering high resolution mode, especially if we wish to leave a previously drawn shape on the screen. In order to signal whether we wish to clear the screen, we will make use of a second flag called CLRFLG. The flowchart shows how these two flags will be used within the machine code routine.

Let's now consider the relatively straightforward task of accessing memory blocks of 256 bytes or less using a machine code loop. The following piece of code places the number $03 into each location from the BASE address to BASE+255 (a total of 256 locations in all) using absolute indexed addressing (see page 196).

```
      LDY $00
      LDA $03
NEXT  STA BASE,Y
      DEY
      BNE NEXT
```

Note that using this technique BASE is accessed first, but the rest of the block is accessed from BASE+255 downwards to BASE+1. Our program calls for us to access more than one 256-byte block of memory, and in order to do this we must use another form of addressing known as post-indexed indirect. This method uses the zero page to calculate addresses anywhere in memory. Most of the zero page is used by the Commodore 64's operating system, but there are a few free bytes set aside for use by the machine code programmer. Two such bytes are 251 and 252 ($FB and $FC). In this method of addressing, the computer assumes that the lo-byte of the address is held in the zero page location specified and the hi-byte is held in the next zero page location. Thus, an instruction such as STA ($FB),Y, where $FB and $FC contain $00 and $20, and Y contains $04, calculates the required address as follows:



STA ($FB),Y
$2000 + $04 = $2004

In order to access an entire 256-byte block of memory, a similar method to the one just described could be used. The power of this method of addressing lies in our ability to access the hi-byte of the BASE address. Incrementing this hi-byte by one means that the BASE address has actually been increased by 256 (i.e. to the start of the next block of memory). We can apply this technique to the tasks of placing colour information into the normal screen area and clearing the bit map area.

The screen area runs from $0400 to $07E7. This means that it consists of three blocks of 256-bytes and a remainder of $E7 bytes. The block of Assembly language labelled "Colour Screen Area" on page 339 makes use of post-indexed indirect addressing to place the colour information into each byte. Use is made of the variables SCBLO and SCBHI — the lo- and hi-bytes of the normal screen start address — and SCBLK and SCREM — the number of 256-byte blocks and the remainder, respectively. PTR is the zero page location used to store the lo-byte of the base address.

CALCULATING PIXEL POSITIONS

The second part of our machine code routine is concerned with the calculation of the bit in the bit map area corresponding to a given (x,y) co-ordinate. For the high resolution screen,

Hi-Res Routine

This performs three different tasks. It sets up the colour screen information, clears the bit map area, and sets and resets the hi-res register bits – dependent on the state of the flags, HRSFLG and CLRFLG