## CREATE PRE-SHIFTED SPRITES WITHOUT MISSING LINK

What has always been a pain for ST users is the ST's sixteen-pixel boundary problem. This means that any graphic has to be copied in multiples of sixteen pixels. For example, if you used the command screen copy to put a piece of the picture on screen then it will round its x coordinates to the nearest sixteen pixels. Look at this example.

screen copy 5,10,10,100,100 to physic,10,10

The command would ignore the value of 10 for the x coordinate and place it on screen at x coordinate 16. Always round up to the nearest step of 16, EG: 16,32,48,64, etc. This can be awkward if you need your piece of graphic to be in its exact place.

The only way to overcome this problem is by pre-shifting. This means making several copies of the graphic with a smaller pixel gap in between. Try this example…

```
10 key off : curs off : flash off : hide on : mode 0

20 dim SP$(16)

30 for X=1 to 16

40 sprite 1,X,16,1 : put sprite 1 : wait vbl : sprite off

50 SP$(X)=screen$(logic,0,16 to 16,32) : cls

60 next X

70 rem Show It

80 logic=back

90 for X=1 to 16

100 cls logic

110 screen$(logic,X,16)=SP$(X) : wait 5

120 screen swap : wait vbl

130 next X
```

This routine shows how to make pre-shifted sprites. Lines 10 to 60 make sixteen copies of sprite one and the rest of the routine moves it across the screen at a step of one pixel at a time. Each image of sprite one is caught in the 'screen$' command, each one captured in its own different

position moved one pixel across further than the last one. Might you, sixteen images in memory can cost a lot of memory so we can make fewer images, this means the pre-shifted graphic moves in bigger steps but takes up less memory. The pre-shifted graphic can move in either 1,2,4,8, or 16 pixels across the screen. In other words, we are cutting down the 16-pixel box into smaller equal size boxes. We can work out how many images we need and how many steps we need to move in like this.

16/STEPS=?

So, we take the number 16 which is the boundary and we divide it by the number of steps or pixels we want the pre-shifted graphic to move and we get the answer to how many copies we need to make of it. For example, if you wanted to move the sprite across the screen at four pixels at a time then you could try this sum.

16/4=4

This means you would have to make four copies of the sprite. Try other sums but remember 'STEPS' must be either 1,2,4,8, or 16.

Now let's imagine we have a block of a picture 100×100 pixels in size and we wanted to place it at coordinates 36,0 on the screen. Using screen copy would place the block at 32,0. Therefore we need to screen copy the block to a dummy screen at coordinates 32,0 and scroll it along to 36,0. To do this we need to use the Def Scroll command like this.

```
10 key off : curs off : hide on : flash off : mode 0

20 reserve as screen 5 : rem Dummy screen

30 screen copy 6,0,0,100,100 to 5,32,0 : rem copy block from bank 6

40 def scroll 1,32,0 to 132,100,4,0 : rem Set block to scroll 4 pixels

50 logic=5

60 scroll 1 : wait vbl

70 ink 0 : bar 32,0 to 35,100 : rem Get rid of picture trail

80 logic=physic

90 A$=screen$(5,0,0 to 320,100) : rem Capture pre-shifted block

100 rem merge it onto screen at right place
```

```
110 screen$(logic,0,0)=A$

120 A$="" : erase 5 : stop
```

So, what this routine is doing is putting our block on a sixteen-pixel boundary in a bank and scrolling it along from X Co-ordinate 32 to X Coordinate 36. Then we can merge it on screen, this takes it from its exact position and puts it on screen at its exact position.

So that's it, with a bit of work you can have a large block moving across the screen like a sprite. This is how to do it all if you don't have the missing link or extra extensions. In the missing link, you can set up your pre-shifted sprites using the MAKE program and you can position blocks anywhere using the ppsc command from the Extra extension.