

## MISSING LINK MAKE PROGRAM

The missing link doc file talks about different kinds of banks for things such as bobs, joeys, blocks, etc. But how do we make those banks in the first place? The answer is simple. On the missing link disk is a file called MAKE.BAS which allows you to convert sprites to other formats.

The missing link uses a new type of sprites called bobs. These are called pre-shifted sprites. The normal STOS sprites work like this. To get past the problem of the ST's sixteen-pixel boundary the sprite has to be pre-shifted. This means so many copies of the sprite have to be made and each one is scrolled by one pixel to the right until there are sixteen copies of one sprite image made before it goes on screen.

This all has to be done before the sprite can be moved or placed on the screen to position it anywhere correctly. This takes a fair bit of processor time which explains why sprites are often jerky and slow. So the routine goes and makes sixteen copies, place the sprite on the screen, make another sixteen copies, place the sprite on screen at the next pixel, and so on. But there is a better faster way of doing this.

The difference between STOS sprites and pre-shifted sprites is that pre-shifted sprites are pre-shifted once and held in memory whilst STOS sprites are pre-shifted as they move across the screen which means they take up less memory than pre-shifted sprites.

Even though STOS sprites make sixteen images of themselves we can decide how many images of pre-shifted sprites are made. We can make either 16, 8, 4, 2, or 1 image. Might you the fewer images you use the more they jerk when moved. This can be fixed by moving them at different steps. So, we need to calculate how many pixels to move an image. So, if we wanted our sprite to move in steps of one pixel we would need to make sixteen images, one for each of the sixteen pixels of the boundary. This is worked out like this.

$$16/1=16$$

We want to move the sprite at one pixel at a time so we divide it by sixteen and we get the answer sixteen. This means we have to make sixteen images of the sprite for it to be moved by one pixel at a time. Now let's suppose we wanted to move it by two pixels.

$$16/2=8$$

This has now halved the size of the image because we're now moving the image at two pixels at a time so only half of the images have to be made. Look at these other examples.

$16/4=4$  : Move sprite at 4 pixels so make 4 images of each sprite.

$16/8=2$  : Move sprite at 8 pixels so make 2 images of each sprite.

So, all we are doing is simply taking the number of pixels we want the sprite to scroll and dividing it by sixteen to find out how many images we need to make. Note that if your sprite is to move up and down the screen only then we only need to make one image as the sixteen-pixel boundary only applies to moving across the screen.

But how do we make these images in the first place? Well, that's where the MAKE program comes in. First, load it up and you are presented with menus covering different convert options. The one we are interested in at the moment is the one to convert our STOS sprites to pre-shifted sprites (bobs). From the LOAD menu click on SPRITES and load your sprites. Next, go to the MAKE menu and click on MAKE BOBS. Here we see three options.

MAKE BOBS

IMAGES

QUIT

Before we convert the sprites to bobs, we need to choose the number of images we need of each one. Position the mouse pointer over the word IMAGES and press the left mouse button. You will now see the word IMAGES with a number next to it and the word SPRITE above that with its image number next to it. Move the mouse pointer over either word and click on either mouse key. Notice how the left and right mouse buttons step through each number of each option.

Select sprite one and image four by clicking on SPRITE till its number says 1 and on IMAGES till it says 4. This means we have set MAKE to make four images of sprite one. If we wanted to make four images of all the sprites then we can simply click on IMAGES with both mouse buttons. Note that if we now step through the sprites, we see they are all set to be converted to four images each.

Now all the images are set we can proceed to make them into bobs. Click on EXIT from the IMAGE selection screen and click on MAKE BOBS to start

the pre-shifting. Notice how MAKE shows the sprites pre-shifting on-screen while it's doing it. How long it takes depends on how many images are being pre-shifted.

When the program stops, click on EXIT to go back to the main menu. Our sprites are now converted to bobs and all we have to do now is save them. From the SAVE menu click on BOBS and save them under their new name making sure the extension is .MBK. It's important to remember that the more images made of each sprite the bigger the bank is so make sure there's enough room on the disk. The MAKE program has just turned the sprite bank into a bob bank for use with the bob commands. We can now use these bobs in a routine.

```
10 key off : flash off : hide on : curs off : mode 0
20 rem First load the bob bank into memory bank 5
30 load "bobs.mbk",5
40 rem Place bob one on screen and move it along by 4 pixels
50 logic=back : XBOB=10 : YBOB=100
60 repeat
70 wash logic,XBOB-10,YBOB-10,XBOB,YBOB
80 bob logic,start(5),0,XBOB,YBOB,0
90 screen swap : wait vbl
100 XBOB=XBOB+4 : until XBOB>300
```

This routine loads our BOB bank into bank five and the rest of the routine zooms it across the screen. See how fast and smooth it moves? Notice also that unlike sprites the bobs can be placed into any bank and they can either be loaded in every time the routine is run or it can be left in there without loading all the time. The reason for WASH and SCREEN SWAP is because unlike sprites the bobs don't clear the trail behind them. Notice also that WASH is set to clear ten pixels behind and over the sprite to ensure no trails are left.

## **JOEYS**

Joeys are only allowed one colour while bobs can use the full sixteen colours. Useful for single colour sprites such as bullets and there are faster as well. You can convert them in the same way as bobs.

## **WORLD BLOCKS**

This option will convert our sprites to world blocks so they can be used by the world command. Same method as converting sprites to bobs.

## **LANDSCAPE BLOCKS**

This option is the same as creating world blocks only that it converts sprites to landscape blocks for use with the landscape command. Note that there's no IMAGES option for this option, this is because landscape only moves its blocks up and down the screen so the sixteen-pixel boundary doesn't exist here so there's no need for pre-shifting. Just click on MAKE BLOCKS and save them.

## **TILES**

Used by the tile and moziac commands. This option works the same as the bob option only that it creates tiles.

Note that these last four options all need to be saved with the MBK extension, like BOBS they are converted into special banks so they can be used by their commands. They are loaded in the same way as bobs.

## **PICTURE**

This option makes up a picture of your sprites, to activate it you must first load the sprites then from the MAKE menu click on PICTURE, the program will then place your sprites on screen in order one after the other and make up a picture. You can then save the picture. The sprites must be 16×16 in size.

## **DIGIBANK**

Normally you can play a sample using the digiplay command but should you be used to using STOS maestro then you may want to put a load of samples into one bank then this is where this option comes in. Let's look at each option, first from the MAKE menu click on DIGIBANK and you'll be presented with a new options screen. Note that you can either select a number or click on the option.

## **LOAD SAMPLE**

Allows you to load a sample into the bank.

## **SAVE SAMPLE**

Unlike STOS maestro's accessory, this option allows you to save a sample out of the bank and on disk as a stand-alone sample.

### **LOAD DIGIBANK**

When you've saved a bank of samples, it becomes a digibank so the digiplay command can understand it. This option allows you to load the previously saved bank for editing.

### **SAVE DIGIBANK**

Saves out a digibank for use with the command.

### **CLEAR DIGIBANK**

Clears the digibank in memory, all samples are lost.

### **PLAY SAMPLE**

Choosing this option will take you to another screen, position the mouse over the number, and press either mouse key to step backward and forwards through the samples in memory. To play it press space and enter the speed. Press space to play it and the sample will play in a loop. Press space again for the main menu.

### **(UN)SIGN SAMPLE**

This option will take you to the play screen. From here select the sample you wish to sign/unsign and press space for the main menu.

### **EXIT**

Takes you back to the main menu.

### **PLAYING A DIGIBANK SAMPLE**

Once you've created your digibank and saved it you can play it using the digiplay command like this.

```
digiplay 1,start(5),SAMPLE,10,1
```

The parameter SAMPLE is normally used for the length of a raw sample. In this case, it ranges from 0 to 49. If the size is less than fifty then the parameter SAMPLE is used for the number of the sample to play between 0 and 49. If SAMPLE equals more than 49 then the data in bank five is assumed to be a raw sample but if it's less than fifty then it's assumed to be a digibank. Here are two examples.

PLAY A RAW SAMPLE AT SPEED 10

```
digiplay 1,start(5),32000,10,1
```

PLAY A SAMPLE FROM A DIGIBANK AT SPEED 10

```
digiplay 1,start(5),4,10,
```

The Make program takes care of all your converting needs for the missing link extension. One thing I would recommend is that you pre-shift your sprites and blocks at four pixels as this is a good combination of speed and smooth movement.