STACKING A MEMORY BANK

As you know, there are only fifteen memory banks that you can use in STOS. Yet some games have more than fifteen pictures or music files overcoming the fifteen-bank limit. This is done by simply sticking all the files into one bank on top of each other. This is called Stacking a Bank.

Let's say we have five pieces of chip music that we want to save along with the program in one bank and call each piece when we need it. Well first to get them all in the bank we need to add up the total length of the files and reserve a bank to this length. If you put your five selected tunes on a disk then type 'dir' you should get a list like this.

Drive A:

MADMAX.MUS 4200

KILLING.MUS 2100

CIRCUS.MUS 8244

ALEC.MUS 4774

STOMP.MUS 8000

Here we have first the filenames followed by the file length, the file lengths need to be added up and then a bank reserved…..

4200+2100+8244+4774+8000=27318

10 reserve as work 5,27320+100

Note how the value in line 10 is slightly higher than the calculated figure. This is because we must always reserve a round figure for the length, we then add 100 bytes just to make a bit more space in the bank. Now we can add more lines to this routine to load the music files into the bank.

20 bload "MADMAX.MUS",start(5)

30 bload "KILLING.MUS",start(5)+4200

40 bload "CIRCUS.MUS",start(5)+4200+2100

50 bload "ALEC.MUS",start(5)+4200+2100+8250

60 bload "STOMP.MUS",start(5)+4200+2100+8250+4780

70 rem SAVE BANK

80 bsave “M_BANK.DAT”,start(5) to start(5)+length(5)

The first file has loaded into the large bank starting from the start of the bank so it has taken up 4200 bytes of the bank which is the length of the first file (MADMAX.MUS). To load the second file in we have to make sure it slots in after the first one so we load it into the bank starting at the position where the first file ends. The next file has to load in and slot in the bank after the first two so we add the values of the first two files to give us the starting position where to load the third file.

So basically, we are loading in each file after the other adding up the values of the previous files to find the start position of the present file in the large reserved bank. Rather than have all this value+value bit we can add up the values into one length like so.

20 bload “MADMAX.MUS”,start(5)

30 bload “KILLING.MUS”,start(5)+4200 : rem Length of first file

40 bload “CIRCUS.MUS”,start(5)+6300 : rem Length of files 1 and 2

50 bload “ALEC.MUS”,start(5)+14550 : rem Length of files 1,2 and 3

60 bload “STOMP.MUS”,start(5)+19330 : rem Length of files 1,2,3 and 4

Note it’s also important that file lengths that are not even must be rounded up to the nearest even figure for the bank to stack properly.

To use the stacked bank in our program we just reserve a bank to the full length and bload it in. Now the easiest and quickest way to play the music is like this. First, we need two arrays that will hold the start address of each file, and the length of each file. The music can easily be played by one line, saving a load of IF statements.

10 reserve as work 5,27320+100

20 bload “M_BANK.DAT”,5

30 dim MUS_ST(5) : rem Reserve array for start addresses

40 dim MUS_LE(5): rem Reserve array for file lengths

50 MUS_ST(1)=0 : MUS_ST(2)=4200 : MUS_ST(3)=6300 : MUS_ST(4)=14550 :

MUS_ST(5)=19330

```
60 MUS_LE(1)=4200 : MUS_LE(2)=2100 : MUS_LE(3)=8250 : MUS_LE(4)=4780 :

MUS_LE(5)=8000

70 input "Choose tune to play (1 to 5)";PL

80 if PL=0 or PL>5 then goto 70

90 rem Play chosen music file

100 A=musauto(start(5)+MUS_ST(PL),1,MUS_LE(PL))

110 wait key

120 A=musauto(0,0,0) : goto 70
```

You can use the same method with packed pictures. In your game you just load in the bank of stacked binary packed pictures and call them up in the same way……for example.

```
10 reserve as work 10,50000

20 bload "PICS.PAC",10

30 dim SCR_ST(4)

40 SCR_ST(1)=4000 : SCR_ST(2)=3200 : SCR_ST(3)=6500 : SCR_ST(4)=9000

50 for X=1 to 5

60 unpack start(5)+SCR_ST(X) : wait 50

70 next X
```

So, there you have it. With this method, you can have as many binary files in a bank as memory permits, and put an end to the bank limit.