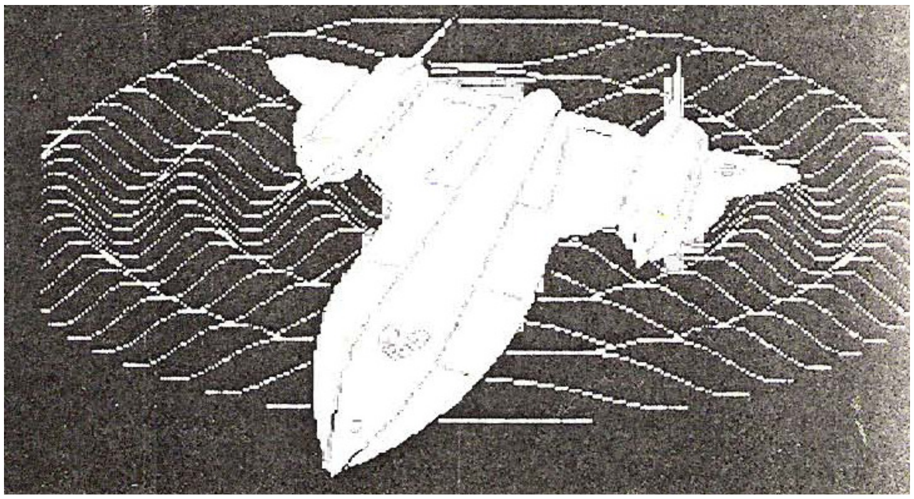
Print ‘n’ Plotter

SCREEN MACHINE

INSTANT MACHINE CODE TECHNIQUES TO IMPROVE
YOUR GRAPHICS PROGRAMMING AND SAVE MEMORY



PRINT ‘N’ PLOTTER SCREEN MACHINE

PRINT ‘N’ PLOTTER SCREEN MACHINE is an invaluable graphics utility for the 48K ZX Spectrum. When used in conjunction with PRINT ‘N’ PLOTTER PAINTBOX (or any other graphics hardware or software) it presents a wide range of extended facilities for both the novice programmer and the professional!

It has been produced after many requests from programmers who wish to have immediate access to instant machine code routines and save time and valuable memory space whilst producing professional results for use within their own BASIC or machine code programs.

PRINT ‘N’ PLOTTER SCREEN MACHINE offers a complete package of such facilities which will make your graphics programming simpler, easier … and spectacular!

And if used together with PRINT ‘N’ PLOTTER PAINTBOX SOFTWARE and the PRINT ‘N’ PLOTTER JOTTER PACKAGE it will give you practically every facility to produce graphics and text on your ZX SPECTRUM to a standard you never thought possible!

WITH SCREEN MACHINE YOU CAN…

… * COMPRESS Screen graphics to cram even more into memory
… * COMPILE TEXT to save memory and give fast machine code access to all text, graphic characters and UDG’s.
… * CREATE easily recallable Multiple Screen Files automatically
… * FLIP SCREEN ANIMATION of Multiple Screen Files are made easy
… * ENLARGE screen sections in 2x steps.
… * REDUCE the whole screen in 2x steps.
… * RECOLOUR the screen graphics globally or selectively.
… * MIRROR the screen image left to right.
… * RELOCATE images to any part of the screen with block or Hi-Res scrolls.
… * SAVE processed Screen Graphics to tape or Microdrive for use in your own programs.

LOADING SCREEN MACHINE

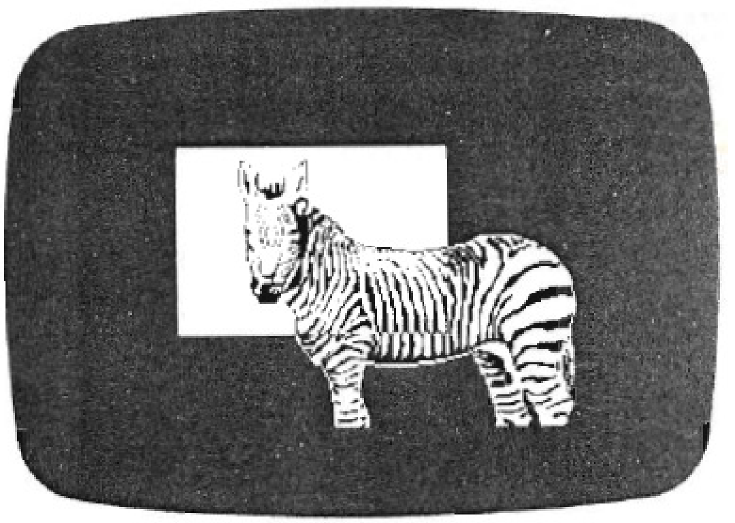
Side 1 of SCREEN MACHINE contains three separate programs and an Index/Loader.
LOAD "" (ENTER) will load the title page and index. Stop the tape when prompted.
The Index gives three options:-

1. **SCREEN MACHINE ONE**
Screen Graphics Compressor (loading time 43 secs)
2. **SCREEN MACHINE TWO**
Screen Graphics Processor (loading time 48 secs)
3. **STRING MACHINE**
Text Compiler (loading time 40 secs)

Select the required option with keys 1-3 and start the tape again. The selected program will load and run. Switch off the tape.

If you keep a note of your tape recorder counter, you can go straight to the required program in future. They are called "sm1", "sm2" and "st" should you wish to by-pass the Index.

Details of how to adapt SCREEN MACHINE for use with Microdrive are available in the appendix of this booklet.



SCREEN MACHINE ONE

Screen Machine One is a utility for Compressing Screen Graphics, Creating Multiple Screen Files and Flip Screen Animation.

You may already known six full 6912 byte screen$ will fit into a 48K Spectrum with only 308 bytes left for a basic program but if you limit the number of Screen$ to 5 then a more useful 7K becomes available for programming.

There are, however, a number of ways of saving memory space for pictures or program.

1. For instance, using only a part of the screen obviously helps, and as the Spectrum screen conveniently splits into three sections, you could have 15 third screens!
2. If you restrict each picture to one INK and one PAPER colour, you don't need to store the attribute file which means you can save 768 bytes on each full Screen$ or 256 bytes from a one third section!
3. Another way to save memory is to store the pictures in a compressed form and expand them again on printing!

SCREEN MACHINE ONE allows the use of any combination of these techniques and organises Screen Graphic loaded into it into a Multiple Screen File which can be saved to tape or Microdrive and is easily recalled from within a users' program.

THE SCREEN COMPRESSOR

Most Screen Graphics contain areas of empty space, ie bytes that are zero. Instead of storing all those blank spaces, SCREEN MACHINE ONE counts them and replaces them with a number. On recall, all the zeros are replaced and the original picture is restored.

The effectiveness of the compression depends on the solid/void relationship of the subject. The most memory efficient images are third screen or monochrome line drawings without large solid areas. For instance, our Zebra is a full screen without attributes and this compresses to 2475 bytes – saving more than 4K!

Because of all the counting of zeros that has to be done, compressed Graphics print slightly slower than normal but unless very fast animation is required, it is quick enough for most purposes – and certainly faster than BASIC!

HOW TO START

The main menu will present you with 4 Options

1. Catalogue
2. Store
3. Load
4. Save

Let's take these options in order of possible use and see how you can start to use SCREEN MACHINE ONE to advantage.

Obviously the first thing you should do is to LOAD in existing graphics for Compressing and storing.

Therefore choose option 2 (STORE) and load from tape a previously SAVE'd SCREEN$ (which you may have produced with PAINTBOX)

Please note that only SCREEN$ files may be loaded into SCREEN MACHINE ONE at this stage, not machine code SCREEN FILES which may have been produced with PAINTBOX

Of course, these SCREEN$ files will be automatically stored as machine code by SCREEN MACHINE ONE – but let's take the facilities in order of use

When your SCREEN$ has successfully LOADED, it will remain on screen with the two report prompts.

These are **C** (to change the present mode)
K (to keep the present mode)
And a notation of the mode in which the SCREEN$ was stored

The existing mode depends on how the SCREEN$ has been previously saved the following table lists the modes which may be of interest but there is no need to remember them

	NORMAL + ATTS	NORMAL - ATTS	COMPRESSED + ATTS	COMPRESSED - ATTS
FULL	3	2	35	34
TOP	5	4	37	36
MID	9	8	41	40
BOT	17	16	49	48

MODE 3 is therefore normal full screen with attributes and requires 6913 bytes to store it. The extra byte is a one byte header that identifies the mode used and instructs the recall routine how to treat the file

If you wish to keep your screen graphics in the present mode you will obviously choose the K option and the following options will be presented

C press key C and report of the number of bytes that the file will need in the present mode will be shown.
S STORE. Stores the Screen$ at the next available memory location

WARNING. DO NOT ATTEMPT TO SAVE A SCREEN$ IF IT DOES NOT FIT BELOW 65535. USE COUNT IF YOU ARE UNSURE.

M Returns to main menu.

If you choose S (Store) the SCREEN$ will be stored and the CATALOGUE page will appear - Please note that CATALOGUE can always be called from the main menu (Option 1).
A description of the facilities available from the CATALOGUE page is as follows.

CATALOGUE keeps a record of all the stored Screen$ and gives their location, length, mode and two numbers which have to be poked to direct the reprinting process.
The total memory space available for Screen$ is 35235 bytes and the very maximum Number of Screen$ is 72 but this is very unlikely as they could only be 489 bytes each.

A FLASHING WARNING "MEMORY" APPEARS IF THERE IS LESS THAN 6920 BYTES LEFT FOR STORAGE.
You should COUNT (key C) the bytes of your next STORE.

Let's assume however that you want to change the present mode to save memory space or only save part of the Screen ...

Instead of pressing K (for keep), press C (for change).
You will then be prompted with the following options:

1. FULL	Stores the whole display file.
2. TOP	Stores the top 8 lines 0 to 7.
3. MID	Stores line 8 to 15.
4. BOT	Stores line 16 to 21. 22 and 23 are not used (see Appendix')

For example, let's choose to save only the top third of the Screen by pressing Key 2.
Immediately you have two further options.

S SCREEN ONLY	Stores without attributes.
A + ATTRIBUTES	Stores them and the display tile.

So let's choose with attributes by pressing A.
- now you have the option of storing in NORMAL or COMPRESSED form!

N NORMAL Normal Screen$ file where speed of recall is important (for animation, etc.)

C COMPRESSED Minimises the amount of memory needed.

To save the maximum memory you may now choose the COMPRESSED option (see note about COMPRESSING earlier in this chapter).
Choose option C.
Now you will have the COUNT and STORE options - or a chance to cancel the whole thing by going to the main menu!
As an exercise COUNT the number of bytes by pressing C.
Obviously with one third Screen and compressed you have saved an enormous amount!
Simply press S to automatically store your graphics and return to CATALOGUE.

You can, of course, carry on storing graphics to the extend of the memory available, but once complete you will need to save the complete file for use in your programs.
This couldn't be easier! Just choose option 4 (SAVE FILES) from the main menu and follow instructions!
The entire file will be SAVED as machine code, complete with recall for your programs. However, certain procedures have to be met in order to use these successfully in your programs, so please read carefully the following instructions:

USING MULTIPLE SCREEN$ FILES IN YOUR OWN PROGRAM
Ramtop must first be lowered by CLEAR 29699 and the complete Screen$ file loaded in.
To recall any image to the Screen$, POKE 23728, I: POKE 23729, h: RANDOMISE USR 30000. I and h come from the CATALOGUE listing.
When you save the Recall Code and Screen$ File from your own program, SAVE "filename" CODE 30300, length: where length is (top Screen$ location plus length of top Screen$) - 30300.

RELOCATING OF SCREEN$ FILES
The machine code routine that recalls the pictures to screen resides at 30000 and the lowest file location is at 30300. Relocation of individual Screen$ is possible provided that the Recall Code (30000-30299) is not moved or overwritten.
Depending on the length of your BASIC program, extra Screen$ can be relocated under 30000.

You will need to know the start location and length of the file from the Catalogue. The following short program will move the bytes. **OLD** is the current start location, **NEW** the new one, and **LEN** the length of the file.

For n = O to **LEN:** POKE **NEW** + n, PEEK (**OLD** + n): NEXT n.

The values for the new POKES are calculated as:

I = **NEW** - INT (**NEW**/256) * 256
h = INT (**NEW**/256)

Then RANDOMISE USR 30000 will call the newly moved file to the screen.

ERRORS. SCREEN MACHINE ONE

Care should be taken to load the correct SCREEN$ and SCREEN$ FILES.

If you load a wrong SCREEN$ or SCREEN$ FILE by mistake or BREAK the program, do not use RUN as this will clear the CATALOGUE, instead, use go to 100. USE RUN only if you want to start from the beginning again.

HIGH MEMORY RELOCATION OF SCREEN$ FILES

If you only want to store one or two Screen$ Files in memory and want more program space, they can be relocated quite easily using the following technique.

Save the Files as normal from within Screen Machine One on to tape keeping a record of the Start Locations and Lengths of each File.

Calculate how much room you need for them in your program by adding the length of each File, adding 600 and subtracting this from your highest available memory location (usually 65535 if not using UDGs).

Lower Ramtop to protect the code by CLEAR (new location - 1) and LOAD "filename" CODE new location.

Now the first 300 bytes of the relocated code are not required as they hold data of the old locations and lengths so you can move Ramtop up 300 bytes to reclaim this much more program space by CLEAR new location + 299 if you wish.

The Recall Code now starts at new location + 300 and the first Screen$ File starts at new location + 600.

The new I and h values to redirect the Recall Code can now be calculated by,

I = f - INT (f / 256) * 256
h = INT (f / 256)

Where f is any Screen$ File start location.

FOR EXAMPLE:

To relocate two Screen$ Files originally at 30300 and 37213 and having 6913 bytes each to the top of memory:

New location = 65535 - (6913 + 6913 + 600) = 51109
CLEAR 51108: LOAD "filename" CODE 51109
CLEAR 51408 to reclaim 300 bytes.
The new File One starts at 51109 + 600 = 51709
I = 253 and h = 201.
The new File Two starts at 51109 + 600 + 6913 = 58622

I = 254 and h = 288.

After POKEing 23728,I and POKEing 23729,h

RANDOMISE USR new location + 300 (51409) recalls the Screen$.

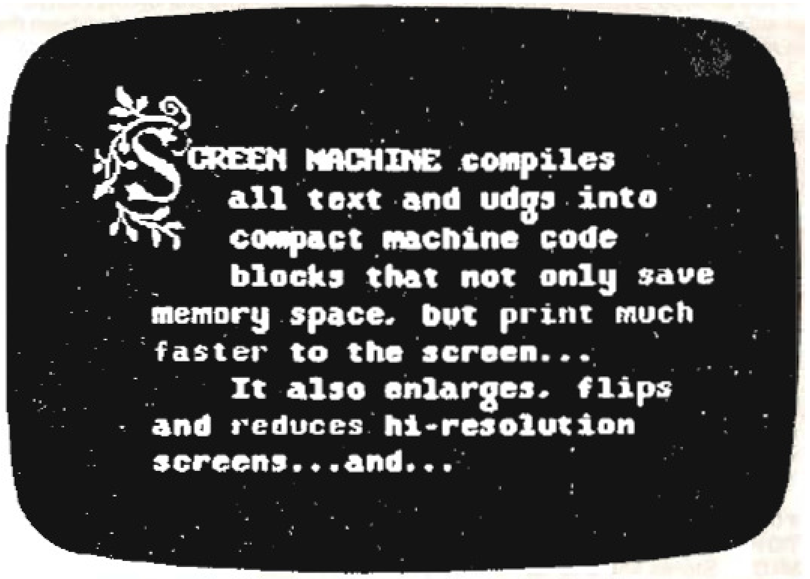
SAVE your new Screen$ File by:

Save "filename" CODE new location + 300, total file lengths + 300.

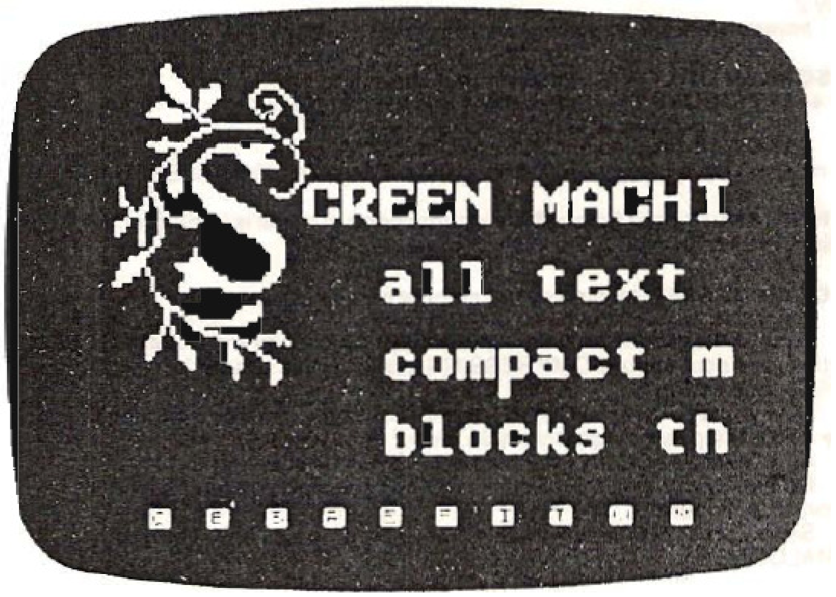
The final option available from SCREEN MACHINE ONE is the facility to re-LOAD a previous SAVED MULTIPLE SCREEN FILE into the program for additions.

This is option 3 from the main menu.

Follow the prompts and the CATALOGUE will show the originally SAVED screen and locations.



```
SCREEN MACHINE compiles
   all text and udgs into
   compact machine code
      blocks that not only save
memory space, but print much
faster to the screen...
      It also enlarges, flips
and reduces hi-resolution
screens...and...
```



```
SCREEN MACHI
   all text
   compact m
   blocks th
```

SCREEN MACHINE TWO

SCREEN MACHINE TWO is a program that manipulates previous drawn Screen$ Files is a number of useful ways.

The program starts with a menu of five options, the first being to load Screen$ Files from tape into permanent memories 1 and 2. These remain intact throughout the operations.

For instance, if you have two previously SAVED SCREEN$ Files (produces with PAINTBOX or other graphics software/hardware) then LOAD both into the program by choosing options 1 and then 2 from the main menu.

Once loaded, your graphics are held in two separate memory locations in order to recall them anytime.

Now, go to DRAWING BOARD (option 3 from the main menu) and you will be presented with a complete range of options along the bottom of the screen.

Make yourself familiar with these options before proceeding - they are:

C. CALL. Calls stored Screen$ to view. Screen$ 1 and 2 are those loaded from tape and Screen$ A and B are to store processed Screens for subsequent saving to tape. Unused memories have a built-in NO FILE PRESENT message.
S. STORE. Stores the current Screen in Memory A or B.
B. BLEND. Calls a Screen$ from memory A or B so that it superimposes on the current screen. Attributes are NOT blended.
E. ENLARGE. Enlarges a portion of the screen indicated by W. (WINDOW) by a factor of two. It also enlarges the attributes proportionally. Use the SCROLLS to line up the desired section in the WINDOW.
R. REDUCE. Reduces the whole screen by a factor of two. Reducing a hi-resolution screen requires a certain amount of compromise and not all subjects will give a satisfactory reduction.

Obviously it is not possible to halve the size of a screen pixel so what we have to do is discard alternate lines and close up the remainder. This effectively reduces the resolution of the image and can lead to loss of information if important lines are discarded.

To remedy this, the screen lines can be thickened prior to reduction. This can be done horizontally (H) or vertically (V) or both, the best result being decided by trial and error.

The attributes cannot be usefully reduced so reduced screens have to be monochromatic. Colour can be added back at a later stage with the I. INK facility or by reloading the picture back into PRINT 'N' PLOTTER PAINTBOX for retouching.
F. FLIP. Gives a mirror image of the screen and attributes.
I. INK. Gives the option of changing the whole screen to a new INK, PAPER and BRIGHT combination or can SELECT particular combinations of INK, PAPER and BRIGHT and change only occurrences of these.

Care should be taken to identify the combinations correctly or nothing will happen. BRIGHT modes are selected by pressing CAPS SHIFT along with the INK colour.
W. WINDOW. Inverses a frame around a window which will be ENLARGED if the E. key is pressed. It only operates whilst the key is pressed and is also useful for centering or positioning scrolled screens.
T. THIRDS. Inverses alternate third screen sections whilst the key is pressed for checking the edges of the screen.
M. MENU. Returns the program to the Menu for Saving stored Screen$ A and B or for Loading further Files into memories 1 and 2.

SCROLLS, USING CURSOR KEYS 5, 6, 7, 8. These are optionally character scroll with attributes or, together with CAPS SHIFT, pixel scroll without attributes. This permits the relocation of images on the screen.

When a picture is scrolled off the screen, the part that goes off the edge is lost and the space on the opposite side becomes blank. (INK 7 : PAPER 0). This effect can be used to isolate or mask an image or to delete parts either temporarily or permanently.

If the colours of an element are important, be careful when using pixel scrolls as the display file will get out of register with the attributes

As an exercise to familiarise yourself with the program facilities, follow these steps with a SCREEN$ you have loaded into MEMORY 1.

1. CALL DRAWING BOARD by pressing key 3 from the main menu.
2. Press C (for CALL).
3. Press 1 (to call stored SCREEN$ 1 to the screen)
4. Press F (to see how easy it is to reverse the picture left to right).
5. Press F (to revert to the right way round!).
6. Press W and in the centre of the screen you will see a rectangular area which is the "enlarge" window.
7. Using the cursor keys (without SHIFT) locate a part of your drawing in this window (pressing W occasionally to check alignment).
8. Press E (for enlarge) and the window area will be enlarged 2x with attributes
9. Press S (for Store) and then press A to store this enlarged portion in MEMORY A.
10. Press R (for Reduce) and then R again to reduce back to original size - but in monochrome.
11. Press I (for INK) and then press W (for Whole Screen). Choose new INK and PAPER COLOURS as prompted.
12. Using Cursor keys move this to the top left hand corner - check screen area by pressing T (for thirds).
13. Store this in Memory B by Pressing S and then B.
14. Now (with cursor keys) move to extreme bottom left
15. Now press B (for Blend) and press A (for Memory A)
    - what you should have on screen is identical graphics one in white and the other coloured.
16. To colour the white version, press I (for INK) and choose S (for Selected)
    - specify OLD INK as 7 (white) and the OLD PAPER that exists on screen
    - then choose NEW INK and PAPER and see the 'white' version change
17. Now press S (Store) and then A (to store this in MEMORY A).
18. Scroll across to extreme right of the screen (check by pressing T) and then press B (Blend) and A (Memory A) to get another white set on screen!

Used in various combinations, SCREEN MACHINE TWO offers some quite stunning effects.

If you want to make each picture a different colour, prepare a coloured chequerboard pattern with PAINTBOX or a small basic program and produce a Screen$ from it on tape. If this is loaded into Memory 2, your multi image screen can be superimposed on to it and the colours changed with I. if you wish.

Horizontal and vertical repeats can be coloured from within SCREEN MACHINE TWO by Storing the original image at the edge of the screen in Memory A, scrolling it over a bit, colouring it, Blending in a new image, colouring it, Blending in a new image, colouring it, scrolling it a bit more, Blending in another repeat, colouring it, and so on remembering that the blank area that comes in from the edge of the screen is always INK 7: PAPER 0. IMPORTANT Remember to Store the finished picture in Memory A or B if you want to Save it to tape

PROCESSING TEXT

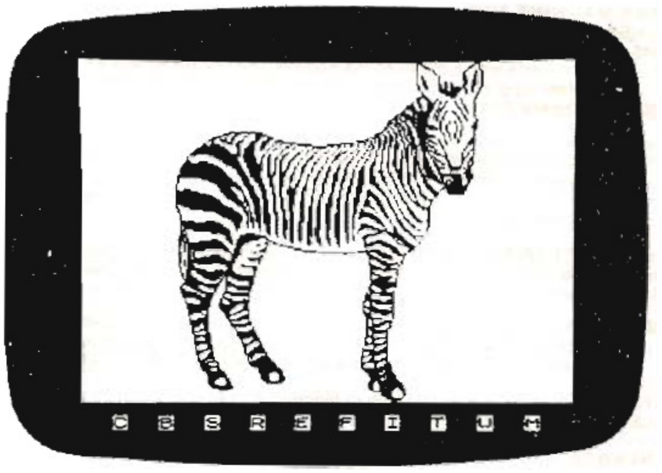
Text can be enlarged to provide heading or titles. The text should be aligned as required in W. WINDOW with scrolls. The image can then be enlarged as many times as you like and re-coloured if necessary.

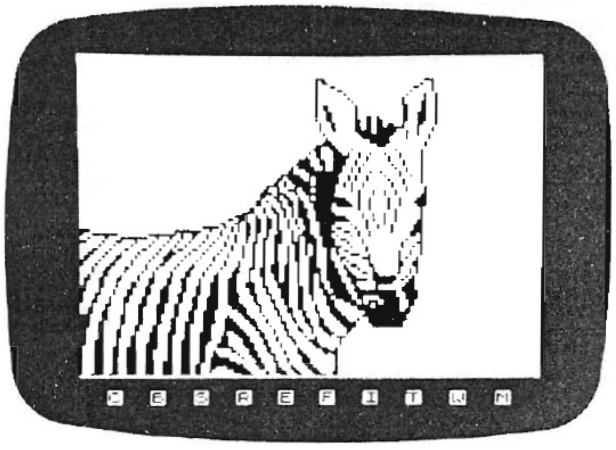
Another useful trick to make type bolder is to Store it, scroll across a pixel line and blend it back with itself. **THIS CAN PROVIDE EXPANDED, CONDENSED AND DROP-SHADOW EFFECTS.**

SAVING SCREEN$ TO TAPE

Options 4 and 5 of the Main Menu will dump Screen$ stored in memories A or B to tape with a filename.

Remember that processed Screen$ can be reintroduced into PAINTBOX for modification or into SCREEN MACHINE ONE for compression.





STRING MACHINE

STRING MACHINE is a very powerful utility that Compiles all text, graphic characters and UDGs, along with their attributes, into machine code blocks. Each is instantly recallable to the screen with just one RANDOMISE USR command. Apart from looking much more professional, considerable saving of memory and speed of printing can be achieved.

On loading from tape the Menu offers:

1. **CATALOGUE.** A list of string that have been created together with their locations and lengths. V. allows you to VIEW individual strings or all of them if "all" is entered. Note that the individually viewed strings do not have a prompt to return to Menu as you may be printing to the bottom two lines normally reserved for messages. Press any key to get the Menu.

   A copy of the CATALOGUE can be sent to a printer with P. DO KEEP A RECORD OF THE CATALOGUE DETAILS OR YOUR WORK WILL BE USELESS.

   The maximum number of strings that can be Catalogued is 48. Do not try to store more that this at one go.

2. **CONSTRUCT STRING.** This is done using lines 5 to 100 of the program itself. A string "a$" is created which contains all the text. screen positions and attribute commands that are required. How to do this is explained later. When the program is RUN, the string is printed and an option to COMPILE or AMEND given.

   If COMPILE is chosen, the option to print to the TOP SCREEN (lines 0 to 21) or BOTTOM SCREEN (lines 22 and 23) is offered. An indication of the lowest available start address is then prompted. You can choose this or any higher location that you wish. The CATALOGUE is then displayed to confirm the entry.

3. **SAVE TO TAPE.** Dumps all the compiled strings to tape with the prompted File Name, Location and length calculated from the CATALOGUE information giving (lowest location) and (last string location plus its length - lowest location).

4. **CLEAR TEXT.** Clears all the lines of text from the listing so that you can construct a new string.

USING STRING MACHINE

The text to be compiled is entered into the program listing in lines 5 to 100. All other lines are suppressed for clarity.

First, change the paper colour to suit the program into which the text is to go with a direct command e.g. PAPER 0 - ENTER. It is best to make the border a contrasting colour to aid alignment of the text at this stage.

The simplest string that can be entered is something like,

```
5 LET a$ = "HELLO"
```

When the program is RUN, HELLO will appear at the top left of the screen. Pressing A. to AMEND will return you to the listing. Now type, 10 Let a$ = a$ + "SAILOR" and use the command RUN.

The strings in lines 5 to 10 have been added or concatenated together.

Now try changing line 5 using the normal basic editing techniques to,

5 LET a$ = CHR$ 22 + CHR$ 5 + CHR$ 10 + "HELLO".

WARNING. EDITING LINES OTHER THAT 5 to 10 MAY CRASH THE PROGRAM

On RUNning you will see that this is the same as PRINT AT 5, 10; "HELLO SAILOR". A fuller explanation of using CHR$ can be found in chapters 14 and 15 of the Spectrum Manual but here is a short list of the most common CHR$ commands:

CHR$ 13 = NEW LINE (ENTER)	
CHR$ 16 = INK	and must be followed by + CHR$ (0 to 9) to indicate colour choice.
CHR$ 17 = PAPER	followed by + CHR$ (0 to 9) to indicate colour choice.
CHR$ 18 = FLASH	+ CHR$ 1 for on or CHR$ 0 for off.
CHR$ 19 = BRIGHT	+ CHR$ 1 for on or CHR$ 0 for off.
CHR$ 20 = INVERSE	+ CHR$ 1 for on or CHR$ 0 for off.
CHR$ 21 = OVER	+ CHR$ 1 for on or CHR$ 0 for off.
CHR$ 22 = AT	+ CHR$ (0 to 21) + CHR$ (0 to 31) to indicate screen locations.
CHR$ 23 = TAB	+ CHR$ (0 to 31) (to indicate column location) + CHR$ 59 (semicolon)

If printing to the two bottom lines of the screen (22 and 2 23) is required, treat them as lines 0 and 1 at this stage, they will appear in the correct place when compiled but remember that when used within your program, they will be overwritten by an error message if some kind of waiting loop or PAUSE is not introduced.

INK and INVERSE commands can be embedded within the quote marks if you would rather do this using INV. VIDEO, TRUE VIDEO, etc

For those who do not know how to embed INK colours into the text, this is done using the following procedure.

Press CAPS SHIFT and SYMBOL SHIFT at the same time to get into EXTENDED MODE. Keeping CAPS SHIFT pressed, now press the appropriate colour number key. Thereafter all text will be in the colour of your choice, until the embedding procedure is used again for another colour.

To return directly to the Menu from the listing simply type:

Let a$ = "menu" : GOTO 100.

If you wish to CLEAR all text and strings from memory then type.

LET a$ = "clear" GOTO 100.

As an indication of the memory that basic text takes up consider our Index Page. The basic program would be:

```
    5 PRINT AT 0, 3; INK 3; BRIGHT 1; "S C R E E N  M A C H I N E"
    10 PRINT AT 6, 6, INK 6, BRIGHT 1, "1 SCREEN MACHINE ONE", AT 8, 9; INK 5;
"Screen$ Compressor"
    15 PRINT AT 11, 6;  INK 3, BRIGHT 1; "2 SCREEN MACHINE TWO"; AT 13, 9, INK
5; "Screen$ Processor"
    20 PRINT AT 16, 6; INK 3; BRIGHT 1; "3 STRING MACHINE", AT 18, 9; "Text
Compiler"
    25 PRINT 0; AT 1, 4; INK 4; FLASH 1; BRIGHT 1; "Select Program to Load"
```

This requires about 510 bytes of memory. Compiled with STRING MACHINE, it needs just 253, and prints must faster!

If compiling text using UDG characters it is best to locate these in memory **after** LOADING STRING MACHINE and when in the text listing mode. Location of UDG's within the UDG memory area will, of course, limit the area allowed for compiling text, but as it is a tiny amount and located very high in memory this should present no real problems.

It is also possible for the dedicated programmer to locate all 4 BANKS of UDG's from PAINTBOX and use these when compiling - see noted about location and RANDOMISE USR calls in the PAINTBOX instruction booklet

USING THE COMPILED STRINGS IN YOUR OWN PROGRAM

Ramtop must be lowered to protect the machine code from basic by CLEAR (lowest location - 1) and the string file loaded in from tape by LOAD "filename" CODE.

A RANDOMISE USR call to any string start location will than print that string

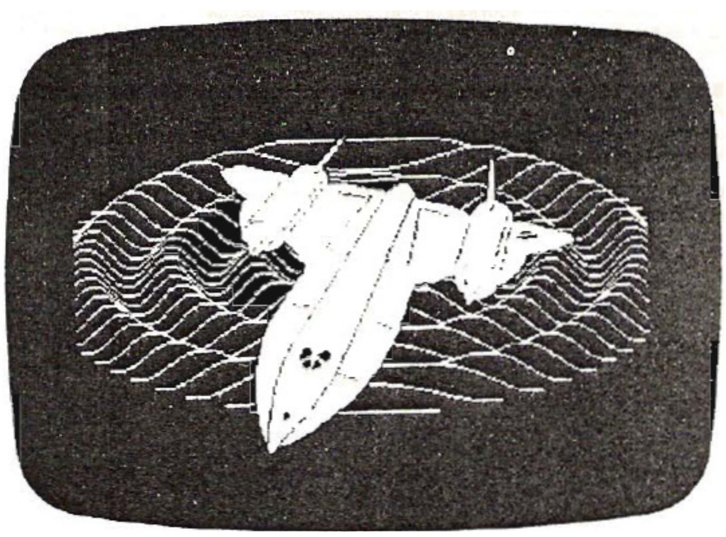
RE-LOCATION OF STRINGS

Strings can be re-located in memory if necessary but will require a few POKES to redirect the machine code to point to the new addresses.

Suppose you want to move a 100 byte string from 35000 to 28000. Lower Ramtop to 28000-1 by CLEAR 27999 then enter; FOR n = 0 to 99; POKE 2800 + n. PEEK (35000 + n); Next n.

Then to redirect the machine code to point to the new text address which is always (location + 16) or 28016 in this case, POKE (location + 6), (location + 16) - INT (location + 16/256) * 256 and POKE (location + 7), INT (location + 16 / 256). RANDOMISE USR 28000 will now print the string.

This process has to be repeated for each string in turn or a short basic program could be written to do it automatically.



DEMO PROGRAM

This is a demonstration of some of the facilities available in SCREEN MACHINE. If you BREAK the program and study the listing you will see what is being done.

100. **TEXT SCREEN.** This is a combination of a small line illustration and a page of text that might be found in an illustrated adventure program. Although saved as a Compressed Screen$ in this particular instance, it could equally well have been a compressed top third screen$ and Compiled String File using UDGs. Line 120 actually calls the Screen$.

200. **PLANE.** This was drawn with PAINTBOX and BLENDED, using SCREEN MACHINE TWO, with a program-generated mathematical figure. Various colour combinations were tried using the I. Colour Change facility and the combination loaded back into PAINTBOX to retouch out some unwanted lines in the cockpit area.

300. **ZEBRA.** this is a particularly efficient compression as it is monochrome and has lots of blank space around it. It compresses to only 2475 bytes against a normal 6912 if saved as a normal Screen$. Line 320 calls the Screen$ and line 330 the words.

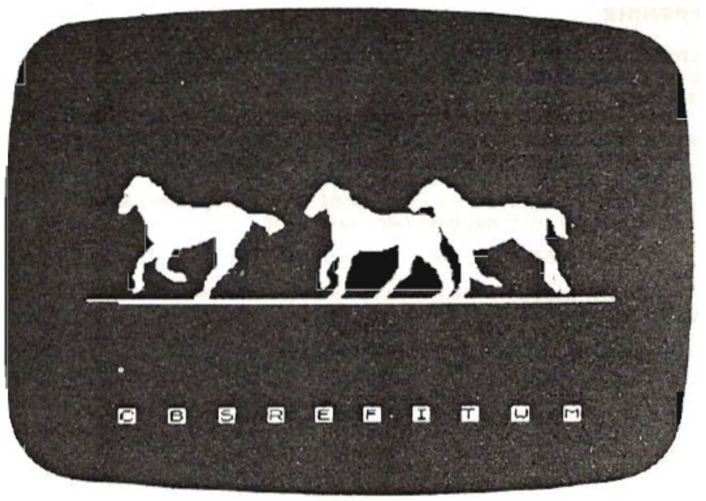
400. **HORSES.** The twelve stages of animation were drawn with PAINTBOX from a series of cine stills. These were enlarged photographically and traced on to sheets of clear acetate. These were stuck to the television screen and an outline traced with the PRECISION PLOTTER facility Fill was then used to make the solid shapes.

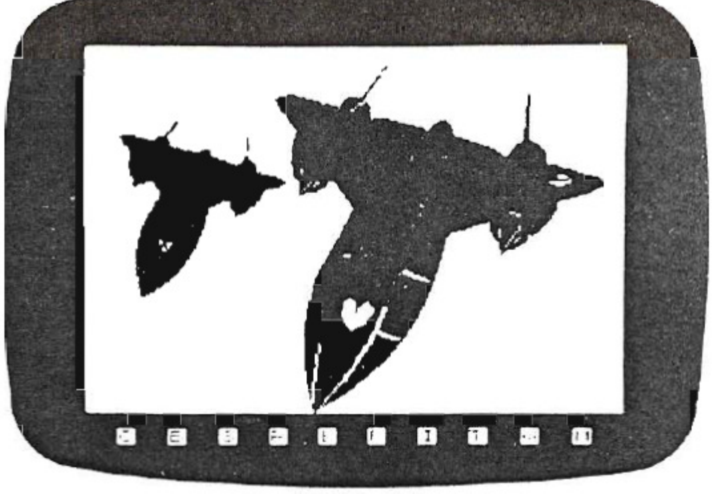
The twelve resulting Screen$ were then processed using SCREEN MACHINE TWO to give three horses per frame. Horse drawings one and five were loaded into Memories one and two. Horse one was scrolled 10 positions to the left and stored in Memory A. Horse five was then scrolled two steps right and Blended Horse one from Memory A. Both were then stored in Memory A and Horse drawing number nine loaded into Memory one. he was scrolled 9 steps to the right and Memory A Blended in. One complete frame with three running horses in different positions were then stored in Memory B and dumped to tape.

This whole procedure was copied eleven more times to produce the 12 three horse Screen$.

The twelve whole Screen$ were then loaded into SCREEN MACHINE ONE and stored as 12 one third Screen$ without attributes and not compressed taking 2049 bytes each.

The Screen$ are called using the FOR-NEXT loops in lines 420-460. The POKES to 23728 and 23729 are READ from DATA statements in lines 500-530. The PAUSE 3 in line 450 controls the speed of the horses. Removing this causes the horses to run too fast and increasing it to 10 gives a delightful slow motion effect.





APPENDIX

SCREEN MACHINE ONE and TWO always saves the bottom two lines of the screen as BLACK PAPER and this may not always be desirable.

The use of INPUT "" will clear the two bottom lines from within your program but a more elegant method is to POKE this portion of the ATTRIBUTE FILE with the colours that you want before re-saving your program.

The locations in question are the last 64 bytes of any Screen$ file (provided that the attributes have been saved.) So

```
FOR n = 0 to 63: POKE last - n, colour: NEXT n
```

last is the start location + the length of the file or, more simply, one less than the next file start.

Colour is INK + PAPER * 8 (+ 64 if BRIGHT 1).

MICRODRIVE ADAPTION

YOUR SCREEN MACHINE CAN BE ADAPTED TO MICRODRIVE BY FOLLOWING THE INSTRUCTIONS BELOW.

In order to do so, you must first BREAK into the various programs to call the listings to screen.

This can be done as follows:

INDEX PROGRAM. Use BREAK keys.
SCREEN MACHINE ONE. Call LOAD from main menu and BREAK while LOADING.
SCREEN MACHINE TWO. Call LOAD from main menu and BREAK while LOADING.
STRING MACHINE. Enter text editing mode.

Give direct command: RANDOMIZE USR 34007 to bring listing to screen.

Thereafter proceed as follows:

INDEX. Change lines 800 to 999.

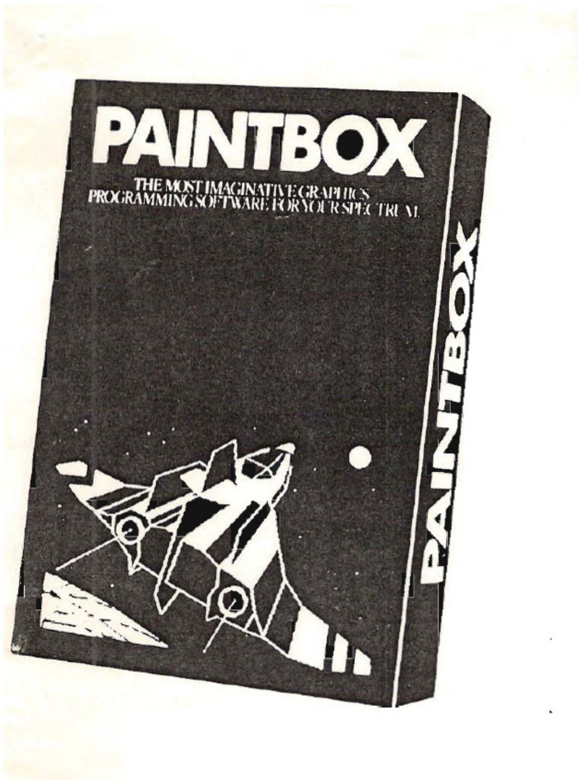
SM1. Change lines 2010, 3020, 4050, 9999
SAVE * "m";1;"sm1" LINE 9999: SAVE * "m"; 1; "sm1c" CODE 28000, 2300

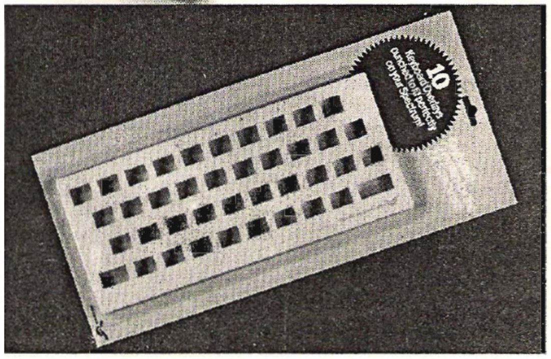
SM2. Change lines 350, 610, 999.
SAVE * "m"; 1; "sm2" LINE 999: SAVE * "m"; 1; "sm2c" CODE 26217, 5000

ST. Change lines 7100, 9999.
SAVE * "m"; 1; "st" LINE 9999: SAVE * "m"; 1; "stc" CODE 33300, 1700.

"DEMO". Change 998, 999.

Please note: **TWO** copies can be put on the same cartridge with identical names by POKEing 23791, 2 before any SAVE command.


PAINTBOX
THE MOST IMAGINATIVE GRAPHICS PROGRAMMING SOFTWARE FOR YOUR SPECTRUM
PAINTBOX



Print'n'Plotter Products

Print 'n' Plotter Products Limited.
19 Borough High Street, London SE1 9SE. Telephone: 01-403 6644.