SUPER CODE

SUPERCODE III

Version 3.5
Operating System

by
Freddy Vachha B.Sc.
1985

THE USER MANUAL

THE ULTIMATE ZX TOOLKIT

FOR THE

SPECTRUM PLUS AND 48K SPECTRUM

Adapted from the Original Supercode
written by F.A.Vachha B.Sc. and V.B.Rumsey

CONTENTS

Published and distributed by Deep Thought Software, 2 Hawthorn Road, Chapel Allerton, Leeds LS7 4PH, United Kingdom.

I ....

INTRODUCTION

1.1 Supercode III is the finest toolkit available for any computer in the world. Its like has not been seen before. It consists of one hundred and fifty two (yes, 152!) purpose written state-of-the-art machine code routines, compact and almost all relocatable, accessed by a powerful BASIC program. The routines can be called either from BASIC or from machine code, either within your program or as direct commands. And, if you have a ZX Interface One, you will be able to transfer Supercode 3.5 to Microcartridge as easy as 1-2-3 (no program changes of any sort at all, the program does it all for you) for the ultimate in convenience (if you don't have an Interface One, all the rest of Supercode's features are still there for you).

1.2 Congratulations on purchasing Supercode III ! It will give you countless hours of pleasure. And there is an added bonus: as you have purchased this copy after 31st December 1984 you are the owner of Supercode III with its Version 3.5 operating system, which gets the very best out of Supercode and is a great improvement (both in speed and in friendliness) over its predecessor 3.0. You can confirm that you have the very latest version by checking the colourful Screen you get while loading the program: it will say 'Version 3.5' on it.

1.3 The routines in Supercode 3.5 (which is Supercode III with the Version 3.5 Operating system) can be broadly classified under two headings:-
   a) UTILITIES give your programming far more flexibility than you would have imagined possible as a BASIC programmer. Indeed, Supercode 3.5 begins where the Spectrum ROM left off! The routines include a variable speed TRACE, ON ERROR GOTO, ON BREAK GOTO (the last two can make your program breakproof and crashproof), SUPERCATALOGUE, several types of RENUMBER (the full version being able to handle all GOTOs, GOSUBs, RESTOREs, LISTs LLISTs,LINEs etc and highlights calculated arguments), Block Moves and Deletes, Line Moves and Deletes, Program Compacting Routines (such as REMkill,Line Contract,Number->VAL"" converter) Variable Search+List+Replace, Program scrambling routines, an ultrafast random number generator, a SCREEN$ compactor/expander, all the routines you will ever need for Channel, Network, Data file and Microdrive manipulation, diagnostics, Tape Header Reader, Program Editing aids like Amend and the Case Converters, and much much more besides....
   b) SPECIAL EFFECTS of all sorts are to be found within Supercode 3.5. Every conceivable scroll, in high (pixel) or low (character) resolution, up/down/left/right/diagonal, the whole screen or a user-definable window, with selectable wrap-around, scroll-off, ripple/shutter effects, with or without attributes. Also, instant changes or swaps of INK, PAPER, BRIGHT, FLASH; instant filling, saving, exchanging, overprinting, merging and inverting of screens, 'impossible' border effects, five sound generators (simulation of whistles,horns,bells,laser zaps,etc), a screen interrogator (useful for finding what is where on a screen, say for detecting collisions in an arcade game) and so very much more..... Any of these routines incorporated in your own programs could make you into an overnight machine code arcade/adventure game programmer, with the colossal speed of Z80 machine code at your disposal. Yes, Supercode 3.5 could well repay your investment hundreds or thousands of times over.

Clearly, Supercode 3.5 is in a class of its own, with over FOUR TIMES AS MANY ROUTINES AS ANY OF ITS CURRENTLY AVAILABLE COMPETITORS. It incorporates many features never previously available to home microcomputer users.

1.4 Supercode 3.5 will work on either the Spectrum Plus or on the 48K Spectrum computer. It is fully compatible with all approved Sinclair accessories (such as the ZX Printer, the Prism modem, ZX Interfaces One and Two, etc). While any properly designed Spectrum accessory (this includes most keyboards, RAM extensions, sound generators and joystick controllers) should also work with Supercode 3.5, we cannot guarantee such operation. In exceptional cases it may be necessary to disconnect such devices before loading Supercode 3.5. This is th fault of the device (which interferes with the Spectrum Operating System) and not of Supercode 3.5. Devices you are specifically warned against include accessories (specially printers) for the U.S. TS1000/2048/2068 computers. They are liable to permanently damage your Spectrum.

1.5 The BASIC program which enables you to access Supercode's 152 routines is there only for your convenience. The program is NOT in any way essential to the operation of the routines: indeed when you use the routines in your own programs or as program preparation aids, it will not be present at all. What the BASIC program does do is largely replace the need for this manual - it is designed like a book, with some 200 screen pages. Each routine has a page to itself, and that page contains all you need to know about the routine: its name,purpose, address in memory, length in bytes, how to call/access it, how to Save it, any customising POKEs to tailor it to your very specific purpose (say in defining the length,width and position of a scren window and the attribute value(s) you wish to be scrolled within it), whether it is relocatable (if not, how to move it), what accessories (eg; Interface One) it requires, and so on. A menu of commands is available at each such page: in most cases, you can actually have an example of the routine working (great fun to watch) fully under program control. This can be repeated as often as you wish. You can also return to that page, to the next page in sequence, to any specified page or back to the Index; print the details to an attached printer; quit to BASIC or Save and automatically verify the routine to either tape or any (1-8) microdrive, with the facility of using either the default filename or of defining your own filename, and all this at just the touch of a key. The program is fully error-trapped using its own ON ERROR GOTO routine.

1.6 Other pages not specific to individual routines give menus with similar options as well as options to Locate routines, to turn the pages of the Index in sequence or jump to a specific page, or have a Demonstration of a compendium of routines, or Transfer the whole of Supercode 3.5 from tape to microcartridge. Do not let the wealth of options put you off - Supercode 3.5 is incredibly user-friendly too.

1.7 Supercode 3.5 has 152 routines but also has the facility to modify these routines for specific purposes. The permutations of possibilities is enormous, running into the hundreds of trillions: Supercode 3.5 is an entire Library on one tape. Used intelligently, Supercode 3.5 can replace the need for mountains of software (assemblers,monitors,disassemblers,debuggers,editors program generators,games designers,sprite generators ,compilers and other miscellaneous toolkits,print utilities,tape utilities etc included) and mounds of reference books.

1.8 Yes, Supercode 3.5 has it all (this documentation was produced using a word processor which utilises Supercode routines !) - I wish you as much satisfaction using it we had writing it. Happy programming !

II .... USING SUPERCODE III

2.1 Supercode 3.5 is supplied to you on a cassette.
Side A: Routines 1 to 76
Side B: Routines 77 to 152

2.2 **To load** Supercode 3.5, first enter RANDOMIZE USR 0 (or simply reset) a Spectrum Plus / 48K ZX Spectrum and play the appropriate side of the cassette. Each side comprises four blocks:

(i) SC3.5-1, a short loader program
(ii) SC3.5-2, a colourful display screen
(iii) SC3.5-3, the actual routines + misc. data
(iv) SC3.5-4.x, the BASIC Access program (x=1 for Side A and x=2 for Side B)

Supercode 3.5 will autostart after about 4 minutes, showing you a diagnostic display (this tells you whether you loaded from cassette or from microdrive, whether or not you have a functional ZX Interface One connected, and the amount (to the nearest K) of working RAM you have in your computer). Ramtop is automatically cleared to 51000.

2.3 The first three blocks on Side A are identical to the first three blocks on Side B. Hence in order to switch from Side A to Side B you can use a shortcut method: Quit the program using the Q option. Enter LOAD "". Start playing the other side of the tape, not from the start but from say 40% or 50% of the way through. The program will now autoload.

2.4 As stated in the Introduction, the BASIC Access program is present only to help you access, tailor and test the routines easily. It is not in any way essential to the operation of the routines. If you wish to load only the routines from tape, proceed as follows:
a>> Enter CLEAR 51000:LOAD "SC3.5-3"CODE and play the tape (it doesn't matter which side) from the start (or from about 15% of the way through to save a little time!). The routines will now load as a block.
If, however, space is at a premium you have two options. Either:
b>> Save the routines you want, and only those routines,to tape (possibly relocating them as well to the top of RAM). There is more information on how to do this later on in this chapter, in the sections dealing with Menu options and Relocation.
or
c>> Alternatively, some space can be saved by ignoring the data etc on routines which is loaded en bloc with the routines themselves. SC3.5-3 is 14535 (hex 38C7) bytes long, starting at 51001 (hex C739) and ending at the very top of RAM,ie 65535 (hex FFFF). Of this area, the first 4351 (hex 10FF) bytes (from 51001 to 55351 (hex D837)) is data used by the Basic Access program. The routines themselves occupy just 10184 (hex 27C8) bytes, from 55352 (hex D838) to 65535. It is hence possible to save space by loading the whole block (using the method in a>>) and then entering SAVE "SC3.5-R"CODE 55352,10184. Record this file on your own tape (and not on the Supercode tape). You now have a recording which contains the 152 routines (mean length exactly 67 bytes) and nothing else, a saving on space of 30% on SC3.5-3 . To use this recording, it is necessary to enter CLEAR 55351 (or any valid lower number, of course) before LOADing.

2.5 Both SC3.5-3 & SC3.5-R will overwrite the UDG area (normally 21*8 bytes starting at 65368 (hex FF58)). If you wish to use UDGs POKE 23675/6 with the 2 byte equivalent (see 2.6) of a more suitable location for them (ensure it is above RAMTOP).

2.6 The term 2-byte equivalent is one that you will come across frequently, both in Supercode 3.5 and in this manual. The 2 byte equivalent of a number N is said to be L (less significant byte) and M (More significant byte) if, and only if, L + 256*M = N, N being in the range 0 to 65535. For example, the 2 byte equivalent of 54321 is 49 (less sig:) and 212 (more sig:) as 49 + 256*212 = 54321. To compute the 2-byte equivalent of a number N (say 54321 again) first divide N by 256, and examine the answer (54321/256 = 212.1914062). The integer part of the answer (212) is the more significant byte, while the fractional part of it (.1914062) when multiplied by 256 and rounded to the nearest integer, gives the less significant byte(.1914062*256=48.9999872 which when rounded gives 49). Alternatively, one can use routine 101 which does the calculation automatically. Lastly, if you are required to POKE X/X+1 with the 2 byte equivalent of a number N, you should POKE X with the less significant byte and POKE X+1 with the more significant byte. For example, POKE 23670/1 with the 2-byte equivalent of 54321 is accomplished by POKE 23670,49: POKE 23671,212 (the lower/first memory location should always contain the less significant byte, please note).

2.7 **The BASIC Access program has a number of menu options,** which are available at different times. Here is a full list:-

A .. **Again:** This option permits you to see the example of any routine another time. It is available after each time you opt to have an example (using E) of a routine.

C .. **Continue** to next routine: This option, available each time you access a routine, moves to the next routine in sequence (ie, if you have been looking at routine 45 it will access the details of routine 46). If you are at the last routine for the Side (76 for Side A, 15 for Side B) and you try C you will be returned to the first page of the Index, except in the case when the program was loaded from Microdrive and you are on 76, in which circumstance SC3.5-4.2 will auto-load from microdrive to allow you to access routine 77.

D .. **Demo:** This option is available both from the Index and after accessing a routine, and allows you to view a rather colourful and noisy demonstration of the operation of a random compendium of routines, after which you are duly returned to the first page of the Index.

E .. **Example:** This option is available after accessing certain routines (an exhaustive list of which is provided in the next chapter) and demonstrates that routine in operation.

I .. **Index:** This option is available as a default when you first load or restart the program, from Save and Locate screens & after accessing a routine. It provides a list of all the routines, showing their number, name and start address (if applicable), and a table of options available (all of which are detailed in this list). Most important features are the pressing of Enter to turn to the next page in the Menu (also refer to J below), and the entering of the no: of a routine (either by typing it in and then Enter, or by first pressing N and then the number and Enter) so as to access that routine. Details of this are given under N.

J .. **Jump:** This permits you to jump to the next page of the Index (there are 4 Index pages on each side, the first 3 containing 20 routines and the 4th 16 routines,and they are numbered 1 - 4). Entering an invalid page number will cause a jump to the next page in sequence (after page 4 comes page 1 again). The J option is available from the Index.

L .. **Locate:** This routine provides details on the relocation of routines, giving on-screen instructions on how to modify some of the routines (namely: 22,23,71,76) classified as non relocatable in order to relocate them. L is available from the Index & after accessing a routine.Refer to 2.8 as well.

M .. **Microdrive:** This option is available after accessing a routine, provided that the routine is non-ROM based and that a ZX Interface One is connected to the computer (note that whether or not Supercode 3.5 was itself loaded from a Microdrive is irrelevant). It allows a routine to be saved either with its default name or with any name of up to 10 characters of your choice, to any Microdrive (1-8, default 1). Immediately after the Save the routine is Verified: if there has been an error restart the program with GOTO 0 and repeat the Save. If the Save is OK you return to the Index.

N .. **Number:** This option is available from the Index as well as after accessing a routine: it allows you to access any routine by entering a routine number in response to the prompt. Valid responses are 1-76 on Side A and 77-152 on Side B, with any other response causing either an error message or a direct return to the first page of the Index. There is one exception, though: provided you loaded Supercode 3.5 from Microdrive, entering a routine number in the range 1-76 from Part 2 (ie, the equivalent of Side B) or in the range 77-152 from Part 1 (the equivalent of Side A) will cause the program (SC3.5-4.1 or SC3.5-4.2, as appropriate) containing the routine you want to load from Microdrive automatically.
N has another feature too: for example, if you have just accessed routine 31 and wish to access routine 38 next, all you need enter is N+7. All mathematical expressions are valid, so to get routine 60 from routine 10 just enter 6*N.

P .. **Printer:** This option is available on each Index page, on each routine page, on the Locate page and on the Save page. Provided a powered and functional ZX Printer is connected to the computer when P is used, a printer dump of the current screen page is obtained.

Q .. **Quit:** This option is available on almost all pages, and allows you to return to BASIC. There are two ways of re-starting Supercode 3.5, as explained below:
You will have noticed that Supercode 3.5 is protected against errors and breaks using routine 65 (ON ERROR GOTO) - you can check this by pressing BREAK while the program is running, by entering N when asked Scroll? or by replying 7G or G7 when asked for a routine number, all of which give different Error Messages (the program even spells out the Error type as defined in the Appendix to the Spectrum manual). To restart the program while retaining this error protection, use GOTO 0. To restart it with the protection disabled, use GOTO 10 instead. ON NO ACCOUNT USE RUN: the program will crash and you will have to reload Supercode.

R .. **Repeat:** This option is available each time you access a routine. It allows you to refer to the details/notes on the routine again (as distinct from A, which gives you the example again) - you can use it as often as you wish.

S .. **Save to Microdrive:** This option is available from the Index as well as after accessing a routine, and gives on-screen instructions on how to transfer Supercode 3.5 (supplied on cassette tape) to Microcartridge. The S operation is valid only when Supercode 3.5 has been loaded from cassette:it is not meant to be a swift method for pirating copies of the program (if you try S from the version transferred to a Microdrive, you will get a Copyright warning message)! The operation of S is as follows:
Load Side A keeping the S key pressed until you are told to release it. This transfers SC3.5-1 & SC3.5-2 to the blank microcartridge you have placed in Microdrive 1. Now reload Side A from the start (not pressing any key) and choose the S option once the main program loads. Press the S key to transfer SC3.5-3 & SC3.5-4.1. Finally, load Side B from the

start (not pressing any key) and choose the S option once the main program loads. This transfers SC3.5-4.2 to microcartridge, so you now have a complete Supercode 3.5 installed on it. Note the following: The original cartridge need not be formatted, it will be auto-formatted while you transfer the first two parts. SC3.5-1 will be transferred with the name 'run'. To load Supercode 3.5 from microdrive either enter RUN on a NEWed / just switched on Spectrum, or enter LOAD *"m";1;"run" .

T .. Tape: This option is available each time you access a non-ROM / Operating System routine. It permits you to Save the routine (with either its default name or any 10 character or less filename specified by you) to tape. Immediately after the Save you MUST verify the routine. If the Verify fails you will be automatically returned to the menu from which you chose T,and you can attempt the Save again if you wish. If the Save is successful you are returned to the first page of the Index.

2.8 You will often wish to **relocate a routine** (ie, locate it at an address other than the one at which it was supplied) either because you want to have more room for BASIC or that you wish to have another machine code routine at the same address. This section explains the relocation of Supercode 3.5 routines. Most of these routines, whose detailed pages do NOT carry the message 'Non-Relocatable - use Locate', can be relocated very simply by using either Method a>> or Method b>> below :-

a>> Save the desired routine to tape or microcartridge. Reset the Spectrum and load it back using LOAD Name$ CODE N or LOAD *"M";1;Name$ CODE N where N is the new address at which you wish the machine code routine to reside. Resave the routine back to tape or microcartridge using SAVE Name$ CODE N,L or SAVE *"M";1;Name$ CODE N,L where L represents the length of the routine. This achieves your objective.

or

b>> Alternatively, to relocate a routine of length L currently located at C to a new address N, use the following program

```
1 LET L=(length):LET C=(Current Address):LET N=(New Address)
2 FOR Z=C+(N<C)*(L-1) TO C+(N>C)*(L-1) STEP (N>C)*2-1
3 POKE Z+N-C,PEEK Z:NEXT Z
```

The routine is now relocated ready for saving.

It is vital to note that the relocated routine MUST be stored above RAMTOP - if necessary, CLEAR a new RAMTOP lower than it. If stored below RAMTOP it may be corrupted by,or itself corrupt, the machine stack,the BASIC program or the variable area, with certainly undesirable, and probably fatal, results !

Another point worthy of mention is that many routines can be tailored/customised by making certain POKEs. Unless the address to be POKEd is in the Systems Variable Area (ie, its address is lower than 23813) it too should be 'shifted' by the same amount as has the routine. For example, if you were to relocate the routine 11 CHR$ MID/LOW LEFT SCROLL from its current location at 64400 to a new address, say 50000, using either Method a>> or b>> above, you would also need to shift the tailoring POKE (POKE 64416 with either 119(for Wrap Around) or 54(for Scroll of)). As the routine has been shifted by 50000-64400 = -14400,the address to be POKEd (previously 64416) must also be shifted by the same amount. The new address to POKE is hence 64416+(-14400) = 50016. POKEs to addresses less than 23813 must NOT be shifted at all.

To relocate routines which are marked 'non-relocatable', it is first necessary to modify them. The experienced machine code programmer will be able to disassemble and then modify them to work from whatever location is required (except 77,78,120 & 125 which by their very nature are almost impossible to freely relocate). However, we have formulated an easy way for the BASIC

programmer to relocate routines 22,23,71 & 76 to addresses that are integral multiples of 256 away from their original addresses (ie, 52683 is an integral multiple of 256 away from 58571 because 58571-52683=5888 which is exactly 256*23). Be warned that this method will NOT work unless this is the case.
Use the following BASIC program:

```
1 LET L=(length):LET C=(Current Address):LET N=(New Address):LET
Q=(Routine number(22,23,71 or 76))
2 IF Q=22 OR Q=23 THEN LET Z1=228:LET Z2=229
3 IF Q=71 THEN LET Z1=235:LET Z2=Z1
4 IF Q=76 THEN LET Z1=231:LET Z2=Z1
5 FOR Z=C TO C+L-1:IF PEEK Z=Z1 OR PEEK Z=Z2 THEN POKE Z,PEEK  Z
+ (N-C)/256
6 NEXT Z:STOP
```

RUN this program with the routine in place: the routine will be modified. Once this is done use either Method a>> or Method b>> to relocate it to address N.

2.9 In the unlikely event that you wish to **MERGE** your own BASIC program with Supercode 3.5 , here is the information you need: Only use line numbers between 2700 & 9400 for your program. Further, there is only 9.1K free on Part 1 (SC3.5-4.1) and 2.2K free on Part 2 (SC3.5-4.2), and when the .8K which must be reserved for temporary Microdrive/Channel files is deducted from these figures, it leaves Part 1 with 8.3K & Part 2 with 1.4K. These are the maximum Program (incl Variables) sizes which can be merged with SC3.5-4.1 & SC3.5-4.2 respectively. Lastly, certain variables are reserved for use with Supercode 3.5: their values must on no account be changed/cleared by your program. These variables are (with their value in brackets): D(0),E(1),F(2),G(3),H(4),I(10),J(5),K(25),M(7),O(8),P(6), Q(11), R(.04),S(255),T(24),U(2440),V(2430),W(2600), X(1000), Y(40) and NMAX(?). You are free to use any other variables you choose.

2.10 When you **save** Supercode 3.5 routines for use with your BASIC programs you may wish that the BASIC & machine code could be saved as one block rather than as two or more blocks. There are two ways of doing this:-
Method 1>> This method will work only with tape and not with Microdrive. Let us say you have a BASIC program called GAME which is about 10K long (use routine 104 to find its length). You wish to SAVE it as well as Routine 65, ON ERROR GOTO. As your program takes up 10K, we need to reserve all memory from 24K (start of BASIC area,approx:) to 24K + 10K + 2K(safety area for temporary variables etc) = 36K. Let us then relocate routine 65 so that it starts at around 36000 (say at 36100). To see how to do that refer to section 2.8 . Now enter CLEAR 65535. (Yes, this will result in the code being stored below RAMTOP but we have located it well above the BASIC area with a 2K margin of error, and far far below the machine stack which grows downwards from RAMTOP). Save the program and code as code (!) using SAVE "GAME" CODE 23755,12417 (12417 being the difference between 36100 + 73 (length of routine 65) - 1 = 36172 and 23755 being the start address of the BASIC area when no initialised Interface One is present). Use the method given in routine 116 to make the POKEs (for the code to autostart when loaded back) to it before saving it, if you so wish.
Another possible location for routine(s) is in the Printer Buffer which stretches from 23296 to 23551, provided they will fit and that a printer is not going to be used by the program. If ON ERROR GOTO were to be stored here, the save command would be SAVE "GAME" CODE 23296,12704 (12704 being 36000 - 23296).
Method 2>> This method will work both with tape and microdrives, and is to be recommended over 1>>. It will work with all routines that are freely relocatable (ie, all but 22,23,71,76,

77,78,120,125 & the ROM/Operating System routines). Let us say you wish to save a routine of length L bytes with your program. Use routine 84 REM FILL to create a REM statement of length L as line 1 of your program. Use routine 68 NON-DELETABLE LINE to make the line number 0, which ensures it will be the first line in the program. Use routine 88 LINE ADDRESS to find the address of this line: it will be either 23817+or 23759 (depending on the presence or absence of an initialised Interface One). Add 1 to this value: the result is the address to which you must relocate the routine. With your program in place all the time , relocate the routine using either Method a>> or b>> in section 2.8 . You have now got the routine POKEd into a REM statement which is part of your program. To call the routine, do not assume that the program must remain statically located (with an initialised Interface One attached, it wil not). The address of the routine will be PEEK 23635 + 256*PEEK 23636 + 5, so the call should be RANDOMIZE USR (or PRINT USR or LET L=USR, according to the routine) PEEK 23635 + 256*PEEK 23636 + 5. Further, if a customising POKE was located at Start Address + K, the POKE you use in your program would be POKE (PEEK 23635 + 256*PEEK 23636 + 5 + K), value.
The program may now be saved to tape or microdrive as a program.

2.11 **The 152 Supercode 3.5 routines may be analysed as follows:**

SCREEN SCROLLING :-	1-23,37-40,47,48,67,122.
SCREEN EFFECTS :-	24-36,41,49-54,69,74-76,89,108,118,119,121, 127,133,147,148.
OTHER EFFECTS :-	42-46,77-79,87,102,126,134.
PROG PROTECTION :-	58,59,65,66,80,98,99,114,115,117,123,124.
PROG COMPRESSION :-	64,82,83,100.
PROG MANIPULATION:-	57,60,61,68,70-73,81,84,85,86,88,90,94-97, 103,104,107,116,135.
GENERAL UTILITIES:-	55,56,62,63,91-93,101,105,106,120.
MDRIVE UTILITIES :-	109-113,125,130,131,136-144.
RS232C UTILITIES :-	128,129,145,146,149-152.

These headings are very broad and some routines would fit into 2 or more different categories: here for simplicity each routine has been classified in only one way.
Also note that Routines 109-113,118,125,128-146 and 149-152 all require an Interface One to be attached to the computer. For Routines 109-112,125 and 136-144 at least one Microdrive must be attached as well, or the routines will not work.

2.12 We have seen a lot of things which Supercode 3.5 can do. Here are two things it cannot do: It cannot teach you Z80 Machine Code, and it cannot create programs totally by itself. If you want to learn Z80 the best buy by far is 'Programming the Z80' by Rodnay Zaks. The third revised edition is freely available (it is published by Sybex, ISBN: 0-89588-094-6) and is a good deal better than sliced bread! The Spectrum BASIC manual is excellent too, with the chapters on 'The Memory' and 'The System Variables' absolutely vital reading. By far the best guide to the Spectrum ROM is the definitive 'The Complete Spectrum ROM Disassembly' by Dr Ian Logan & Dr Frank O'Hara (it is published by Melbourne House (Publishers) Limited, ISBN: 0-86759-117-X). I have examined many books on the Interface One but am unable to really recommend any to you.
As to creative ideas, these must come from you. Creative software of any seriousness is some way off yet. But Supercode 3.5 should make things easy for you. I will confine myself to just one example: you want an arcade/adventure with scrolling screens, colourful pictures, random enemy targets as well as a galaxians section. For scrolls you have a gigantic choice. To store and retrieve screen pictures without wasting memory you can use 147 SCREEN$ COMPRESS & 148 SCREEN$ RETRIEVE. It may be

possible for you to draw some of the pictures from within your program itself, using 76 PAINT-FILL to colour it. Random enemy targets can be produced using 55 RND# GENERATOR and 54 SCREEN$ PRINT, while collisions/accuracy of fire can be monitored using 53 SCREEN$ SEARCH. Sound effects for firing is best done with 42 LASER ZAP (with different values POKEd in to tailor it for types of fire) with 44 DUAL-NOTE SOUND-GEN & 46 MULTI-BEEP SIMULATOR to signal the end of the game, progression to a new level, etc. Screens can be very elegantly cleared using 19 SHUTTER LEFT-SCROLL (or 21 SHUTTER RIGHT-SCROLL), possibly preceded by either 18 RIPPLE LEFT SCROLL or 20 RIPPLE RIGHT-SCROLL, called repeatedly, and some calls of 41 ATTR SWOP or 74 FLASH SWOP to simulate full screen explosions. Galaxian effects are best caused with 2 PIXEL DOWN SCROLL with successive reprinting of the lower landscape (29 SCREEN$ MERGE could be handy) or even the sideways scrolling of it (9 CHR$ LOW LEFT-SCROLL or 15 CHR$ LOW RIGHT-SCROLL). Make the character set look more interesting with 79 SCIFI CHR$ SET. Protect the program with 68 NON-DELETABLE LINE and 66 ON BREAK GOTO (or 124 DISABLE BREAK). Save the program with 115 HEADERLESS FILES, check with 80 PROTECT PROGRAM and 123 ANTI-MERGE PROGRAM. If you wish to be nasty use 98 CONFUSELIST. To save space try 100 COMPRESS NUMBERS and 82 CONTRACT PROGRAM. To make it look neat, use 60 SUPER-RENUMBER. That is just a taste of what is possible.......

2.13 Supercode 3.5 routines are meant for use in your own programs, and the authors do not wish to exercise any of their rights under the laws of copyright relating to such use. You are free to use Supercode 3.5 routines without limitation, except in other toolkits (a very necessary precaution, as Supercode 3.5 has many imitators but no equals). Further, if you use these routines in programs which are sold commercially, we require you to acknowledge the fact that you are using Supercode 3.5 routines both in the advertising copy for the program and within the program itself (possibly on the loading screen). I am sure you agree that these conditions are very reasonable. You do not require the permission of the authors to use any of the routines, or pay any royalty to them, provided you comply with the above conditions.

III .... TABLE OF ROUTINES

No:	Routine Name	Address	Length	(1)	(2)
1	PIXEL UP-SCROLL	64001	97	RL	EX
2	PIXEL DOWN-SCROLL	64098	99	RL	EX
3	CHR$ / ATTR UP-SCROLL	3190	N/A		EX
4	PIXEL LEFT-SCROLL	65462	32	RL	EX
5	PIXEL RIGHT-SCROLL	65494	32	RL	EX
6	CHR$ LEFT-SCROLL	64275	25	RL	EX
7	CHR$ TOP LEFT-SCROLL	64300	25	RL	EX
8	CHR$ MID LEFT-SCROLL	64325	25	RL	EX
9	CHR$ LOW LEFT-SCROLL	64350	25	RL	EX
10	CHR$ TOP/MID LEFT-SCR	64375	25	RL	EX
11	CHR$ MID/LOW LEFT-SCR	64400	25	RL	EX
12	CHR$ RIGHT-SCROLL	64425	25	RL	EX
13	CHR$ TOP RIGHT-SCROLL	64450	25	RL	EX
14	CHR$ MID RIGHT-SCROLL	64475	25	RL	EX
15	CHR$ LOW RIGHT-SCROLL	64500	25	RL	EX
16	CHR$ TOP/MID RIGHT-SCR	64525	25	RL	EX
17	CHR$ MID/LOW RIGHT-SCR	64550	25	RL	EX
18	RIPPLE LEFT-SCROLL	64575	18	RL	EX
19	SHUTTER LEFT-SCROLL	64593	18	RL	EX
20	RIPPLE RIGHT-SCROLL	64611	18	RL	EX
21	SHUTTER RIGHT-SCROLL	64629	18	RL	EX
22	PIXEL BOXLEFT SCROLL	58571	112		EX
23	PIXEL BOXRGHT SCROLL	58608	75		EX
24	SCREEN$ FILL	64828	30	RL	EX
25	SCREEN$ STORE	64744	12	RL	
26	SCREEN$ OVERPRINT	64756	28	RL	EX
27	SCREEN$ EXCHANGE	64784	25	RL	EX
28	SCREEN$ INVERT	64809	19	RL	EX
29	SCREEN$ MERGE	63976	25	RL	EX
30	INK CHANGE	64858	25	RL	EX
31	PAPER CHANGE	64883	31	RL	EX
32	FLASH ON	64914	17	RL	EX
33	FLASH OFF	64931	17	RL	EX
34	BRIGHT ON	64948	17	RL	EX
35	BRIGHT OFF	64965	17	RL	EX
36	ATTR FILL	64982	44	RL	EX
37	ATTR UP-SCROLL	65026	55	RL	EX
38	ATTR DOWN-SCROLL	65081	62	RL	EX
39	ATTR LEFT-SCROLL	65204	52	RL	EX
40	ATTR RIGHT-SCROLL	65143	61	RL	EX
41	ATTR SWOP	65256	21	RL	EX
42	LASER ZAP	63950	26	RL	EX
43	UNI-NOTE SOUND-GEN	64647	28	RL	EX
44	DUAL-NOTE SOUND-GEN	64675	31	RL	EX
45	UNI BEEP SIMULATOR	63000	10	RL	EX
46	MULTI BEEP SIMULATOR	63010	24	RL	EX
47	OBLIQUE SCROLL-OFF	63034	17	RL	EX
48	CHR$ DOWN-SCROLL	63051	73	RL	EX
49	CHR$ ROTATE	63163	42	RL	EX
50	CHR$ REFLECT Y-AXIS	63124	19	RL	EX
51	CHR$ REFLECT X-AXIS	63143	20	RL	EX
52	24 LINE PRINTING	ROM	Based		EX
53	SCREEN$ SEARCH	60039	123	RL	
54	SCREEN$ PRINT	63728	49	RL	EX
55	RND# GENERATOR	63777	18	RL	EX
56	BLOCK MEMORY INSERT	63795	11	RL	EX

No:	Routine Name	Address	Length	(1)	(2)
57	BLOCK LINE ERASE	63806	96	RL	
58	CHR$ SWOP	63902	43	RL	EX
59	CHR$ SCRAMBLE	ROM	Based		EX
60	SUPER-RENUMBER	59294	681	RL	
61	LINE RENUMBER	64706	38	RL	
62	DEC->HEX CONVERTER	60595	118	RL	EX
63	HEX->DEC CONVERTER	60713	113	RL	EX
64	REM KILL CONDENSER	60494	101	RL	
65	ON ERRORGO TO	60826	73	RL	
66	ON BREAKGO TO	60899	72	RL	
67	FREE-SCROLLER	ROM	Based		EX
68	NON-DELETABLE LINE	ROM	Based		
69	BORDER EFFECTS	60000	38	RL	EX
70	INITIALISE	63382	108	RL	
71	VARIABLESLIST	60222	185		EX
72	STR$ LIST	63490	154	RL	EX
73	STR$ REPLACE	63644	83	RL	
74	FLASH SWOP	60162	30	RL	EX
75	BRIGHT SWOP	60192	30	RL	EX
76	PAINT-FILL	59136	158		EX
77	RECORD SOUND	65290	28		
78	REPLAY SOUND	65318	32		
79	SCIFI CHR$ SET	57344	768	RL	
80	PROTECT PROGRAM	ROM	Based		
81	APPEND STATEMENT	60407	86	RL	
82	CONTRACT PROGRAM	61400	687	RL	
83	EXPAND PROGRAM	62087	317	RL	
84	REM FILL	58892	244	RL	
85	DATA FILL	63205	177	RL	
86	ANALYSE PROGRAM	62404	129	RL	EX
87	TAPE HEADER READER	62533	286	RL	
88	LINE ADDRESS	59975	13	RL	
89	SCREEN$ GRID	62819	38	RL	EX
90	MONOCHROME PROGRAM	62943	54	RL	
91	ANALYSE MEMORY	62857	86	RL	EX
92	HEXINPUT	65350	112	RL	
93	AWAIT KEYPRESS	60972	24	RL	
94	UPPER-CS STR$	58833	59	RL	
95	LOWER-CS STR$	58774	59	RL	
96	UPPER-CS PROGRAM	58715	59	RL	
97	LOWER-CS PROGRAM	64211	59	RL	
98	CONFUSELIST	58263	135	RL	
99	UNCONFUSELIST	58398	173	RL	
100	COMPRESS NUMBERS	58115	148	RL	
101	2 BYTE CONVERTER	ROM	Based		EX
102	FOREIGN ACCENTS	57176	168	RL	EX
103	MEMORY AVAILABLE	64197	14	RL	EX
104	PROGRAM LENGTH	59988	12	RL	EX
105	RESET	0	N/A		
106	BLOCK MEMCOPY	58683	31	RL	EX
107	BLOCKLINE COPY	61000	400	RL	
108	STAR/RING DRAW	ROM	Based		EX
109	FAST LOAD MDRVE	ROM	Based		
110	SURE SAVE MDRVE	65277	8	RL	
111	MDRVE DIAGNOSIS	ROM	Based		
112	ADAPT PROGRAM	57159	17	RL	

No:	Routine Name	Address	Length	(1)	(2)
113	SURE CLOSE #	57126	33	RL	
114	STOP PROGRAM	ROM	Based		
115	HEADERLESS FILES	57108	18	RL	
116	AUTORUN CODE	ROM	Based		
117	ANTI- COPY PROGRAM	57094	14	RL	
118	ATTR RESET	ROM	Based		EX
119	LOWER SCREEN$ CLS	3438	N/A		EX
120	TRACE VARI-SPEED	56640	450		EX
121	PARTIALCLS	57090	6	RL	EX
122	LOWER UP-SCROLL	65526	6	RL	EX
123	ANTI- MERGE PROGRAM	ROM	Based		
124	DISABLE BREAK	ROM	Based		
125	SUPER-CATALOGUE	55648	992		
126	REACTION TIME	ROM	Based		EX
127	PSEUDOLOAD	1278	N/A		
128	SEND RS232 BYTE	55643	5	RL	
129	RECEIVE RS232 BYTE	55637	6	RL	
130	DESELECT DRIVE	55632	5	RL	
131	SELECT DRIVE	55627	5	RL	
132	KEYBOARDINPUT	55621	6	RL	
133	SCREEN$ OUTPUT	55616	5	RL	
134	PRINTER OUTPUT	55611	5	RL	
135	IF1 INITIALISE	55608	3	RL	
136	OPEN # DATA FILE	55601	7	RL	
137	CLOSE # DATA FILE	55594	7	RL	
138	ERASE MDRVE FILE	55591	3	RL	
139	READ NEXT DATARECORD	55584	7	RL	
140	SAVE NEXT DATARECORD	55577	7	RL	
141	READ RND DATA RECORD	55570	7	RL	
142	READ RND DATA SECTOR	55563	7	RL	
143	READ NEXT DATASECTOR	55556	7	RL	
144	SAVE NEXT DATASECTOR	55549	7	RL	
145	ERASE CHANNEL	55542	7	RL	
146	CREATE CHANNEL	55535	7	RL	
147	SCREEN$ COMPRESS	55430	105	RL	
148	SCREEN$ RETRIEVE	55383	47	RL	
149	OPEN# NET CHANNEL	55376	7	RL	
150	SEND # NET PACKET	55366	10	RL	
151	GET #NET PACKET	55359	7	RL	
152	CLOSE #NET CHANNEL	55352	7	RL	

NOTES
a>> RL in Column (1) implies that the routine is relocatable. If RL is absent, the Locate option may state how to modify routine to make it suitable for relocation.
b>> EX in Column (2) implies that an example of the operation of the routine is provided in the program.

IV .... DETAILS OF ROUTINES

All routines are called by RANDOMIZE USR start address, (the latter being found from the table given in III) except where stated otherwise (eg. 103, called by PRINT USR start address). Also remember that routines 1-76 are to be found in Part One (Side A) and 77-152 in Part Two (Side B).
Note that the table in III contains the actual names of routines (restricted as they are by maximum filename lengths) - the names used here are the expanded forms and the two may hence differ.

1 PIXEL UP-SCROLL
2 PIXEL DOWN-SCROLL
These two routines will scroll the screen up or down one pixel, leaving the attributes unchanged. Use repeated calls to the address specified in the table to the scroll as far as required. By combining these routines with numbers 37-40, joint scrolling of attributes can be done. Define a suitable box, use an attribute value of 63 and call the attribute scroll routine once for every 8 calls of this routine.

3 CHR$/ATTR UP-SCROLL
This routine is in ROM. Also see routine 122.

4 PIXEL LEFT-SCROLL
5 PIXEL RIGHT-SCROLL
Used for scrolling left or right one pixel. Use as routines 1 and 2. POKE start address + 13, with 55 (scroll-off) or with 63 (wraparound) or with 0 (inverse scroll).Also see routines 22-23.

6 CHR$ LEFT-SCROLL
7 CHR$ TOP LEFT-SCROLL
8 CHR$ MID LEFT-SCROLL
9 CHR$ LOW LEFT-SCROLL
10 CHR$ TOP/MID LEFT-SCROLL
11 CHR$ MID/LOW LEFT-SCROLL
12 CHR$ RIGHT-SCROLL
13 CHR$ TOP RIGHT-SCROLL
14 CHR$ MID RIGHT-SCROLL
15 CHR$ LOW RIGHT-SCROLL
16 CHR$ TOP/MID RIGHT-SCROLL
17 CHR$ MID/LOW RIGHT-SCROLL
These routines will scroll the screen one CHR$ square in each direction, leaving the attributes unchanged. Use repeated calls to the address specified in III to scroll as far as required. To scroll attributes call first the routine above, and then one of routines 37-40, after defining an appropriate box and setting the attribute value to 63. For a wraparound scroll, first POKE start address + 16 with 119. To scroll-off, POKE start address + 16 with 54.

18 RIPPLE LEFT-SCROLL
19 SHUTTER LEFT-SCROLL
20 RIPPLE RIGHT-SCROLL
21 SHUTTER RIGHT-SCROLL
These four routines are all pixel scrolls affecting the screen but not the attributes. Ripple rotates each CHR$ about its own axis while Shutter scrolls-off each CHR$ square.

22 PIXEL BOXLEFT SCROLL
This non-relocatable routine pixel scrolls a user-defined box on the screen. The box must, however, be completely within a third of the screen i.e. within the first eight lines on the screen, or the middle eight lines, or the bottom eight lines.
POKE start address + 108, TAB position of the top right-hand

square of the box i.e. the no. of CHR$s from the left-hand edge of the screen to its top right-hand corner.
POKE start address + 109, with 64 if the box to be scrolled is located in the upper third of the display, 72 for the mid third and 80 for the lower third.
POKE start address + 110, width of the box in pixels (i.e. 8 * number of CHR$s).
POKE start address + 111, length of the box in CHR$s.
To scroll attributes as well, refer to routines 37-40. Call the attribute routine once for every 8 calls of this routine. Please note that the above 4 POKEs must be made each time the routine is called. An example BASIC program to do this would be:

```
10 FOR Z=1 TO 225
20 POKE 58679,29:POKE 58680,72:POKE 58681,32:POKE 58682,28
30 RANDOMIZE USR 58571
40 NEXT Z
```

23 PIXEL BOXRIGHT SCROLL
As for 22 except that the scroll is to the right. The increments to the start address are now 71, 72, 73 and 74 and the first POKE relates to the top left-hand corner of the box. Note the routine is non-relocatable.

24 SCREEN$ FILL
Will fill a box on the screen with any CHR$ code.
POKE start address + 1, CHR$ code
POKE start address + 3, box height in CHR$s
POKE start address + 6, box width in CHR$s
POKE start address +4/+7, PRINT AT coordinates for the top left-hand corner of the box.

25 SCREEN$ STORE
Saves a screen in memory.
26 SCREEN$ OVERPRINT
Erases the existing screen and prints the stored one.
27 SCREEN$ EXCHANGE
Swops the existing screen with the stored one.
The three related routines (25,26,27) all use a screen stored above RAMTOP, using 6912 bytes (24*32*8 pixels+24*32 attributes) It may be necessary to CLEAR a new lower RAMTOP. To store a screen from X to X + 6911, you must POKE start address +1/+2, 2-byte equivalent of X.
Incidentally, you can do a SCREEN$ COPY (ie dump) to printer with RANDOMIZE USR 3756.

28 SCREEN$ INVERT
Will invert the colours over the whole screen (ink and paper colours will change at each PRINT position without disturbing the screen).

29 SCREEN$ MERGE
A useful addition to routine 25,26,27. These can be used to simulate animation effects. A screen stored in RAM is merged with the current display. POKE start address +4/+5 with the 2-byte equivalent of the RAM screen's first byte. Attribute values remain unaltered.

30 INK CHANGE
Instantly changes the ink colour over the whole screen.
POKE start address + 1, overall ink colour.

31 PAPER CHANGE
As for ink change, but sets paper colour instead.

32 FLASH ON
33 FLASH OFF
34 BRIGHT ON
35 BRIGHT OFF
These four routines change the specified attribute settings instantly over the whole screen. Contrast these with routines 74-75.

36 ATTR FILL
37 ATTR UP-SCROLL
38 ATTR DOWN-SCROLL
39 ATTR LEFT-SCROLL
40 ATTR RIGHT-SCROLL
For each of these a box can be defined in which the attributes will scroll.
POKE start address + 1, new attribute value
POKE start address +4/+3, PRINT AT coordinates of the top left hand corner of the box.
POKE start address +6, box width in CHR$s
POKE start address +7, box height in CHR$s
On routine 39 wraparound can be achieved by POKE start address + 36, 26. To cancel, POKE start address + 36, 0.
On routine 40 wraparound can be achieved by POKE start address + 43, 26. To cancel, POKE start address + 43, 0.

ATTR SWOP
This routine will search the display area for all characters with a given attribute X, and replace these with a new attribute Y.
POKE start address + 1, X
POKE start address + 2, Y
Refer to ATTR in the index of the Spectrum manual for the explanation of the numbers to use, especially the explanation of the ATTR function itself.

42 LASER ZAP
This makes a futuristic laser sound.
POKE start address + 1, duration.

43 UNI-NOTE SOUND-GEN
This produces a programmable whistle.
POKE start address + 1, frequency
POKE start address + 2, span
POKE start address + 4, duration
POKe start address + 23, 28 (for up) or 29 (for down)

44 DUAL-NOTE SOUND-GEN
This provides two sound channels.
POKE start address + 7, duration
POKE start address + 18, frequency of first note
POKE start address + 27, frequency of second note

45 UNI BEEP SIMULATOR
Replaces the ROM BEEP routine.
POKE start address +1/+2, 2-byte equivalent of the frequency
POKE start address +4/+5, 2-byte equivalent of the duration in units of .02 second

46 MULTI-BEEP SIMULATOR
This can give amazing bell-like effects.
POKE start address + 1, pitch decrement per step
POKE start address + 2, number of notes
POKE start address +4/+5, 2-byte equivalent of initial frequency
POKE start address +7/+8, 2-byte equivalent of duration in milliseconds

47 OBLIQUE SCROLL-OFF
The wierdest scroll of all - call it repeatedly to get the desired effect.

48 CHR$ DOWN SCROLL
For each call of this routine the screen is scrolled down by 8 pixels. To scroll attributes as well, use routine 38 (after defining a suitable box) in conjunction with this routine.

49 CHR$ ROTATE
50 CHR$ REFLECT Y-AXIS
51 CHR$ REFLECT X-AXIS
These three routines operate on any character set stored in RAM. It could be the UDGs, the SciFi CHR$ set described in routine 79, the original CHR$ set copied over from the ROM or any other set devised by you (see the note on the system variable CHARS in the Spectrum manual).
POKE start address +3/+4 (+1/+2 for routine 51) , 2-byte equivalent of the address of the CHR$ to be transformed

52 24 LINE PRINTING
To PRINT lists or text using all 24 lines of the screen, use POKE 23659,0 just before each print instruction and POKE 23659,2 just after. Use PAUSE n if necessary to stop a scroll? message from corrupting the display. If BREAK is pressed while 23659 is set to 0 the Spectrum will crash. If the lower part of SCREEN$ scrolls when being reprinted/updated/amended, use routine 119 to clear it before each print other than the first one.
Alternatively, use PRINT#1; AT K,0; to print on lines 22 & 23 : K=0 for 22, K=1 for 23.

53 SCREEN$ SEARCH
This routine finds the CHR$ code (UDGs included) at the position last PRINTed at. For example, to find what was printed at X,Y use PRINT AT X,Y;:LET L=USR start address. L now contains the required value.

54 SCREEN$ PRINT
This routine PRINTs a CHR$ at any position with any attribute (not necessarily those preset globally).
POKE start address + 4, Ink colour (default 1)
POKE start address + 10, Paper colour (default 6)
POKe start address + 16, Flash (default 1)
POKE start address + 22, Bright (default 1)
POKE start address + 28, Inverse (default 0)
POKE start address + 34, Over (default 0)
POKE start address + 46, CHR$ value
POKE start address + 40/+43, AT coordinates for print position

55 RND# GENERATOR
Call with LET L=USR start address. The routine places a random number in the range 0-65535 in the System Variable SEED: it is accessed by PEEK 23670 + 256*PEEK 23671.

56 BLOCK MEMORY INSERT
This routine inserts a given number from 0-255 into a block of memory.
POKE start address + 1, no: of bytes to be inserted
POKE start address +3/+4, 2 byte-equivalent of address of first byte to be changed
POKE start address + 6, value to be inserted
In the program example, value 200 is inserted into a section of SCREEN$.

57 BLOCK LINE ERASE
This routine can erase series of lines from BASIC programs. Before calling this routine you must:
POKE 23728/9, 2-byte equivalent of the number of the first line to be deleted
RANDOMIZE number of last line to be deleted.

58 CHR$ SWOP
This routine exchanges all occurrences of the CHR$ with code X within a program with the CHR$ with code Y (for a list of CHR$ codes see the chapter on the ASCII character set in the Spectrum manual). UDGs,graphic CHR$s and keywords are all coped with.
POKE start address + 1, X
POKE start address + 3, Y

59 CHR$ SCRAMBLE
23606/7 contains the 2-byte equivalent of the address of the CHR$ set in use. By POKEing these with random numbers, an apparently corrupted CHR$ set is obtained making your program very hard to read. To normalise POKE 23606,0:POKE 23607,60. It is necessary to normalise before any screen printing is done.

60 SUPER-RENUMBER
This routine will renumber all GOTOs, GOSUBs, LISTs, LLISTs, LIST#s, RESTOREs, SAVE...LINEs etc. A list is displayed of line and statement numbers of all calculated (ie,X*10+70) and non-integer (ie,240.6) arguments which need to be manually altered by inspection (structured programs will not have any of these and can hence be renumbered fully automatically). If the argument<>any existing LINE no:,the next valid line no: is used.
POKE start address + 286, interval between lines (default 10)
POKE start address +288/+289, 2-byte equivalent of the number of the first new line (default 10)
If the renumbering would result in line numbers greater than 9999, the interval and first line no: are both set to 1.

61 LINE RENUMBER
A short routine for use where available memory is scarce. It will not renumber GOTOs, GOSUBs etc unlike routine 60.
POKE start address +5/+6, 2-byte equivalent of line interval
POKE start address +8/+9, 2-byte equivalent of the number of the first new line.

62 DEC->HEX CONVERTER
63 HEX->DEC CONVERTER
These two routines auto-repeat. Enter 'Q' to return to BASIC. Only integers between 0 and 65535 (HEX 0 to FFFF) are allowed.

64 REM KILL CONDENSER
Shortens and speeds up your program by deleting all REM statements in it.Use routine 103 to determine the memory saving.

65 ON ERROR GOTO
Call this at the start of your program, with say 1 RANDOMIZE USR start address. On running, errors (other than Interface One errors with the shadow ROM paged in, which cannot be trapped) will not stop the program but will cause a jump to the line number whose 2-byte equivalent has been POKEd into start address +52/+53 (default 9495). This line can contain any error routine you fancy (note that PEEK 23681 gives the error number). Error codes 0,8 and 9 are not trapped by ON ERROR GOTO as they are not really errors but legitimate program stops. Note that after any error the machine stack is reset, so RETURN will not function. Compare this routine with no: 66 . Note that Supercode 3.5 is itself protected using ON ERROR GOTO (hence the Q option !).

66 ON BREAK GOTO
Similar to no. 65, but only covering errors D (Break), H (Stop in Input), and L (Break into Program) with the error code in 23681 again. (note that codes go from 0 to 9 & then A to R, where A=10,B=11 etc). POKE start address +53/+54 with the 2-byte equivalent of the line number to be jumped to when an error occurs. Compare also with routine 124.

67 FREE-SCROLLER
Routine in ROM. To scroll a screen greater than 22 lines long automatically, include the statement POKE 23692,n where n is the number of lines to be scrolled. To scroll forever, include POKE 23692,255 within the loop that generates the screen output.

68 NON-DELETABLE LINE
Routine in ROM. In order to make the first line of your program difficult to delete, enter LET X=PEEK 23635 + 256*PEEK 23636 : POKE X,0:POKE X+1,0. It is now suitable for placing copyright messages in. More difficult to reverse is to make a REM line in the middle/end of your program non-deletable. First find its address Y using routine 88, then POKE Y-4,0:POKE Y-3,0. The program will work perfectly (despite the lines being out of sequence) provided you do not GOTO/GOSUB/RESTORE that line.

69 BORDER EFFECTS
Produces a colourful and eyecatching border pattern.
POKE start address + 6, duration (from 1 to 127)
POKE start address + 20, colour (from 0 to 7)
POKE start address + 29, line spacing (from 1 to 255)

70 INITIALISE
Zeroes all numeric variables/arrays, sets all strings to "" (empty) and fills all dimensioned string arrays with CHR$ 32s.

71 VARIABLES LIST
To display all variables (numeric, string, numeric array, string array & FOR...NEXT loop control variables) used in your program, enter PRINT; : RANDOMIZE USR start address. Useful with routines 70,86 & 103. Note that this routine is non-relocatable.

72 STR$ LIST
A powerful find routine which searches the BASIC program and lists each line containing a specified sequence of characters.
POKE start address +12, number of characters in the string.
POKE start address +9/+10, 2-byte equivalent of the address of the first CHR$ code in the string poked by you into memory (the default is 23296, the start of the printer buffer area)
Any character obtainable from the Spectrum keyboard may be searched for, including keywords, variables and arithmetic operators (+,-,/ etc.). The following BASIC program demonstrates the routine in operation.

```
10 INPUT "ENTER STRING TO BE LOCATED:";N$: POKE 63502, LEN N$
20 FOR Z=1 TO LEN N$:POKE 23295+Z, CODE N$(Z):NEXT Z
30 CLS : RANDOMIZE USR 63490 : PRINT '"NO MORE OCCURRENCES"
```

Note: To enter a keyword (ie,RUN,POKE,SAVE etc), enter THEN followed by the keyword. Backspace the cursor & delete THEN.

73 STR$ REPLACE
Similar to routine 72, but replaces the string found with another string of identical length.
POKE start address + 4, number of CHR$ to be replaced
POKE start address +8/+9, 2-byte equivalent of the address of the string to be replaced.
POKE start address + 69/+70, 2-byte equivalent of the address of the new string.

74 FLASH SWOP
This routine sets every flashing square on the screen to steady & every steady square to flashing. Contrast with routines 32/33.

75 BRIGHT SWOP
As for no: 74, but with BRIGHT. Contrast with routines 34/35.

76 PAINT-FILL
Draw a closed convex figure (the simplest example of which is a circle) on the screen. Plot a point within it and POKE start address + 157, attribute value to be used for filling (ensure that the paper colour matches the surrounding paper or there will be odd boundary conditions).The routine is non-relocatable.

77 RECORD SOUND
78 REPLAY SOUND
These two non-relocatable routines require you to first CLEAR 32767. Call the first routine once you are supplying suitable sound input (from your tape recorder/hifi system) to the EAR input on your Spectrum. You have 5-10 seconds of recording time. Replay is achieved by calling the second routine. which will direct output both to the Spectrum speaker and to the MIC socket, from where you can amplify the signal. Experiment with levels, input sources and sound/music types to optimise sound quality, but expect no miracles. Routine 77 will overwrite all RAM from 32768 to immediately below itself.

79 SCIFI CHR$ SET
POKE 23606/23607, 2-byte equivalent of the start address of the set (which occupies 96*8=768 bytes) less 256. Since the current address is 57344, and 57344-256=57088=0+256*223, the required POKEs are hence POKE 23606,0:POKE 23607,223 .
You will be amazed at the change - POKE 23606,0:POKE 23607,60 to normalise.

80 PROTECT PROGRAM
1) Introduce as line 1 a REM statement and then POKE (PEEK 23635 + 256*PEEK 23636),100. The program will work but it will not LIST (until you POKE (PEEK 23635+256*PEEK 23636),0
2) POKE 23636, 150 to make the program apparently vanish. POKE 23636,92 to make it appear again (it was there all the time !).
3) Use INPUT LINE instead of INPUT - one cannot then erase quotes and enter STOP to BREAK into the program. CAPS SHIFT and 6 will cause a BREAK nonetheless, but this is not well-known.
4) Use routines 59,65,66,98,99,115,116,117,123 & 124.
5) Embed colour control characters (see chapter on Colours in the Spectrum manual) immediately after the line number for the first line of the program. Set INK and PAPER to the same colour. Then change the line number to 0 (see routine 68).
6) Make your program autostart by SAVEing it with SAVE "name" LINE Z, where Z is the first line that will be executed when the program is LOADed. Make the first statements in line Z read:-
LET ERR=256*PEEK 23614 + PEEK 23613:POKE ERR,0:POKE ERR+1,0
The program will now CRASH if you try to BREAK into it (by using BREAK, generating an error, entering n to a scroll? request, replying to an INPUT with STOP or an INPUT LINE with CAPS SHIFT & 6, etc) the computer will reset itself.
7) In the start line of an auto-start program, include POKE 23659,0. The effect will be the same as in 6), but be careful with screen operations within the program - some could be fatal!

81 APPEND STATEMENT
It is very painful having to move the cursor to the end of long program lines: this routine does it all for you. Just move the edit cursor to the desired line as normal,then call the routine.

You will now be in edit mode, but with the cursor at the end of the program line, allowing swift appending of statements and editing of statements near the end of the line.
To speed up cursor movement with long lines, enter POKE 23561,5: POKE 23562,2: POKE 23608,0: POKE 23609,9. The longer the line, the more pronounced is the speed improvement.

82 CONTRACT PROGRAM
This routine will contract your program into as few program lines as possible,by automatically combining statements wherever this was possible without changing program logic. It allows BASIC programs to run faster and occupy less memory, as can be checked with routines 103/104. Contrast this with routine 83.

83 EXPAND PROGRAM
The opposite of 82 CONTRACT PROGRAM, this routine prompts you for the number of the line you wish to expand into all its constituent statements, enabling easy editing. All such lines will have line number 0, so it is necessary to use one of the Renumber routines (60 or 61) on the program afterwards. To expand the whole of your program, respond to the first prompt with just Enter.

84 REM FILL
This routine will prompt you to enter the line number of an empty REM statement you have already created and the number of bytes to be filled into it. A REM statement of the form REM XXXX ..... XXX is then created, ideal for storing machine code (upto 9999 bytes) which can be loaded jointly with a BASIC program. For an example, refer to 2.10 Method 2>> in Chapter II. Also refer to routine 85.

85 DATA FILL
This routine loads data / machine code programs stored in memory into an auto-created DATA statement at line 1.
POKE start address +4/+5, 2-byte equivalent of address of data/ program in memory
POKE start address +1/+2, 2-byte equivalent of the number of bytes to be stored
This routine has a purpose similar to that of routine 84 but is relatively very wasteful of space. For example, if we were to use each routine to store 1000 bytes (all of which are 77), memory consumption would be as follows:-
```
Routine 84: 2+2+1+1 (Line no:,length,Enter & REM) + 1000 = 1006
Routine 85: 2+2+1+1 (Line no:,length,Enter & DATA) + 999(commas)
            + 1000*(2 digits + CHR$(14) + 5 floating pt) = 9005
```
The difference is very significant, the example not being untypical.

86 ANALYSE PROGRAM
This routine displays the number of lines and statements there are in the program. It is useful with routines 71 & 104.

87 TAPE HEADER READER
After calling this routine, start your tape as if to load a program. A full print of header information (name,type,length, address if code, autostart line if any) will be printed on the screen for you to analyse.

88 LINE ADDRESS
Move the edit cursor to the desired program line and then call this routine with PRINT USR start address. You will get the address of the first character in the line. Refer to the chapter on the Memory in the Spectrum manual where line structure is diagrammed and explained in full.

89 SCREEN$ GRID
This routine sets the CHR$ squares on the screen alternately bright & dark (ie, a checkerboard effect). This can be very useful in designing screen layout, calculating PRINT AT and PLOT values, etc.

90 MONOCHROME PROGRAM
This routine removes hidden colour items (other than those within strings) hence saving memory.

91 ANALYSE MEMORY
This routine prints out the address,contents in decimal,contents in hexadecimal and CHR$ value (where printable) of a block of memory. Press N to return to BASIC.
POKE start address +9/+10, 2-byte equivalent of the address from which you wish memory to be analysed.

92 HEX INPUT
For this routine to work routine 63 HEX->DEC CONVERTER must also be in memory. You enter data directly in hex and it is stored in memory where required. To quit, press Q. Note that if you enter >2 hex digits only the last two will be evaluated.
POKE start address +10/+11, 2-byte equivalent of the start address of HEX->DEC CONVERTER (default 60713, its current location in RAM)
POKE 23563/4, address of the first byte of RAM into which you wish to input hex

93 AWAIT KEYPRESS
Call this program with LET L=USR start address within your own program. It waits for a key to be pressed and returns with the CHR$ CODE of the key stored in L.

94 UPPER CASE STR$
95 LOWER CASE STR$
These routines will convert all items within string quotes in the program into upper case and lower case respectively. You must ensure that program logic has not changed (Different responses to A & a as INKEY$/INPUT/INPUT LINE input commands,for example).

96 UPPER CASE PROGRAM
97 LOWER CASE PROGRAM
These routines will convert all items in the program listing (other than those within string quotes, and keywords (which always are in upper case)) into upper case and lower case respectively.
Note that case can be changed from within the program itself using POKE 23658,8 (to Upper) or 0 (to Lower).

98 CONFUSE LISTING
This routine changes all numbers/digits in the program, other than those in REM statements or within string quotes, into a random code to confuse the listing. The program will work perfectly until and unless a line containing a number (made random) is edited, in which case irreversible corruption will occur which makes the program unusable. This routine relies on the Spectrum's way of storing numbers,ie both in visible digit form and 'invisible' floating point form. While the digit form is what you see, the invisible form is what is used for all calculations. This routine alters only the visible form and hence does not affect program execution. However,in any edit all floating point forms are recalculated to make them equal to the digit forms. Hence the routine's effectiveness. Include a copyright REM statement as the last statement in a line

containing many numbers important to your program. Any attempt to delete or change your copyright message using edit will hence make the program inoperable. Also see routine 99.

99 UNCONFUSE LISTING
This routine undoes the effect of 98 CONFUSE LISTING, except in cases of lines which have been irreversibly corrupted for which nothing can be done.

100 COMPRESS NUMBERS
This routine saves memory by storing all numbers as VAL STR$ (ie 23.7 is stored as VAL"23.7") except 0 which is stored as NOT PI. Storage using VAL STR$ slows the program down but saves a lot of memory - 3 bytes per number (if the number has x digits, VAL "number" will occupy 1+1+x+1=x+3 bytes, while stored normally it would occupy x+1+5=x+6 bytes). Supercode 3.5 uses this method to maximise space-utilisation. It also defines all commonly occurring numbers (identified with the help of routine 73) as numeric variables (see 2.9 in Chapter II) and saves these variables with the program - this frees a lot of memory (it is also the reason why trying to RUN Supercode 3.5 is immediately fatal, as would be CLEARing it). Use routines 103/104 to monitor how much space has been saved.

101 2-BYTE CONVERTER
Refer to 2.6 in Chapter II for a definition of 2-byte equivalent. This ROM routine permits instant conversion of any number X from 0 to 65535 into its 2 byte equivalent. Just enter RANDOMIZE X, then PEEK 23670 & PEEK 23671 will give the less significant and more significant bytes respectively.

102 FOREIGN ACCENTS
No, this routine does not speak ze French ! All it does is provide a set of accented CHR$s for use as UDGs (suitable for French,German,Dutch etc).
POKE 23675/6, 2-byte equivalent of start address of routine

103 MEMORY AVAILABLE
This routine, called by PRINT USR start address, prints out the free memory in bytes available to BASIC (ie, the distance from the top of the variables area to the bottom of the machine stack growing downwards from RAMTOP).
A way of saving a BASIC program as code is as follows. Find out the memory available, say M, using this routine. Find RAMTOP by calculating PEEK 23730 + 256*PEEK 23731: say it is R. Now if you enter POKE 23637, PEEK 23635: POKE 23638, PEEK 23636: SAVE Name$ CODE 23552,R-M-23552 you have saved the program as code. When loaded back the program will start at its first line.

104 PROGRAM LENGTH
This routine, called by PRINT USR start address, prints out the length of the BASIC program (ignoring variables).
Incidentally, the fastest way to distinguish 16K & 48K Spectrums is to PRINT PEEK 23733. Its 255 for a 48K Spectrum (or Spectrum Plus) but only 127 for a 16K Spectrum.

105 RESET
This routine simulates a complete power down. It not only does NEW but also clears RAMTOP to its original value, resets all the System Variables and UDGs to their default values, etc.

106 BLOCK MEMORY COPY
This routine moves a block of memory from one location to another. Do not use it to copy BASIC (use routine 107 for that) or your BASIC program will be corrupted.

POKE start address +1/+2, 2-byte equivalent of address of the first byte of memory to be moved
POKE start address +4/+5, 2-byte equivalent of no: of bytes to be moved
POKE start address +7/+8, 2-byte equivalent of the destination address

107 BLOCK LINE COPY
Call this routine and follow the prompts (for no: of first line to be copied, no: of last line to be copied and position the block is to be copied to, in that order) to copy a block of lines from one part of a BASIC program to another. The original lines are not deleted (if they were, the routine would be Move and not Copy) but you can delete them using routine 57. The new lines will all be given line number 0 (but they will be in the right place) so it is necessary to use one of the Renumber routines (60 or 61) immediately afterwards. Note that GOTOs, GOSUBs, RESTOREs, SAVE....LINEs etc within the block of lines will retain their original values, so they may have to be manually adjusted. Any attempt to copy a block to within itself or to have overlapping blocks or to have a line number > 9999 will all result in Error B, Integer out of range.

108 STAR/RING DRAW
Plot a point (stick to the middle of the screen at the start) and enter DRAW X,Y,Z where X,Y are in the range -60 to +60 and Z is >200 - try Z=189*PI . You will be amazed at what you see (a nice ROM bug)!

109 FAST LOAD MICRODRIVE
The ZX Microdrive spends most of its time trying to locate files - actual loading is very rapid. Two ways of saving files to cut down on access/locating times are as follows:-
1) Save >1 copy of the file/program onto the same micro-cartridge, using POKE 23791,X before the SAVE instruction (X being the number of copies wanted). Before using ERASE on such multiply SAVEd files, try the same POKE. Due to unpredictable sector-ordering on the Microdrive, however, only one copy may be erased instead of all, so it is best to check using CAT.

2) When SAVEing a multi-part program (like Supercode 3.5, which, as you know, has 4 parts) VERIFY each part immediately before SAVEing the next part in sequence. The head-positioning this causes ensures that there will be minimal delay between loading the different parts.
Incidentally, it is best not to format ZX Microcartridges just once - five times is much more advisable. Not only does this improve tape mobility/reliability/conditioning but it probably allocates more sectors as being 'good', giving greater storage. One other matter: if you BREAK or reset in the middle of a FORMAT (whether or not the cartridge had been formatted before) you must FORMAT again. Do not use the cartridge without doing this, even it appears to work.

110 SURE SAVE MICRODRIVE
Before attempting a save to microdrive, the Spectrum / Interface One does not check that sufficient memory is free for it to be able to open the necessary channels/maps. If free memory is scarce because of long programs or low RAMTOPs, the computer would crash (with loss of all data) when a SAVE* was attempted. Call this routine before saving long programs to check whether there is sufficient space for a save to microdrive to be safe. If the routine returns with any message other than 0 OK, the SAVE* must NOT be attempted. Instead the program should be shortened and/or RAMTOP raised using CLEAR.

111 MICRODRIVE DIAGNOSIS
1) To check whether or not a functional ZX Interface One is attached, see if an error is produced when an instruction using the shadow ROM (like CLS #) is executed/attempted. This could be used in conjunction with 65 ON ERROR GOTO to trap the error in the instance when no Interface One was attached. If the Interface One is present, this will also page in its shadow ROM.
2) To check whether an Interface One that is connected is also paged in, enter PRINT PEEK 23635 + 256*PEEK 23636. If the answer is 23755, then the shadow ROM has not yet been paged in; if not, it has.
3) To check whether a program just loaded came from tape or from microdrive, enter PRINT PEEK 23787 + 256*PEEK 23788. If the load was from a microdrive, the result printed should be the same as that obtained using routine 104.

112 ADAPT PROGRAM
As was discussed in 2.10 in Chapter II, programs with machine code stored in REM statements written before the release of the Interface One probably will not work when loaded into a Spectrum with the shadow ROM paged in. This is because the program, and hence the machine code in it, will load to a different address due to the extra system variables,channels etc. To correct, run this routine before loading from cassette. It resets system variables to pre-paging values, pages out the shadow ROM and ensures that programs from cassette load to 23755.
Incidentally, to check whether your Spectrum is an Issue 3/3B version or not, PRINT IN 57342. If the result is 191, it is Issue 3/3B. If it is 255, the issue is 1 or 2. If it is neither, then Sinclair have produced a new issue Spectrum.
To make all programs for Issue X Spectrums work on Issue Y Spectrums, precede all IN commands with:
OUT 57342, 191 if X = 1 or 2 and Y = 3 or 3B
OUT 57342, 255 if X = 3 or 3B and Y = 1 or 2

113 SURE CLOSE #
Due to a shadow ROM bug CLOSE # does not always succeed in closing all streams. This routine, however, does.
Incidentally, to disable LLIST and LPRINT include the statement OPEN #3;"s" at the start of your program.
Also, another way of performing OPEN #N (3<N<16) is to use POKE 23754 + 2*N, 19 + 2*N .

114 STOP PROGRAM
Here are ways of breaking into autostart programs:
1) If in BASIC, use MERGE instead of LOAD.
2) If in machine code, use routine 87 to find it's length and start address. Then load it to a different address, typically high in RAM.
3) If a headerless file, disassemble its LOAD routine and find the number, say X, that it loads into DE (LD DE,X). Then use routine 115 with a higher start address and Number of bytes = X. The file should now load and then stop.

115 HEADERLESS FILES
This routine will load a headerless file from tape to any specified address. It will execute the code once loaded, if that is required.
POKE start address +2/+3, 2-byte equivalent of the number of bytes to be loaded
POKE start address +6/+7, 2-byte equivalent of the address to which the first byte is to be loaded
POKE start address +15, 195 if the machine code is to be executed on loading. If yes, POKE start address +16/+17, 2-byte equivalent of the address from which the machine code is to be

executed. If no, you will be returned to BASIC at the end.
Save this routine within the first part of your program (in a REM statement, say) and execute it from within the program.

116 AUTO RUN CODE
To make a machine code program autostart, locate it as low in memory as possible,storing it under RAMTOP (this is because saving a region of memory including the stack can be very tricky and is best avoided). Enter 1 RANDOMIZE USR start address , and then POKE 23637,PEEK 23635: POKE 23638, PEEK 23636: SAVE Name$ CODE 23552,Z-23551 where Z is the address of the last byte of code. When LOADed, the BASIC program will autorun and start the code. Note that this method cannot be used for saving to microdrive.
If a functional Interface One is connected to your Spectrum, you can call a machine code program by having as the first program line 1 REM call address 2 POKE 23582, 27 . Now PRINT £4 will be able to call the machine code.

117 ANTI-COPY PROGRAM
Save all the parts of your program other than the first with the header of other programs (to do this just involves manipulation of your tape recorder). To make the system work, call this routine from within part 1 of your program. Immediately after call routine 115 which should contain all the relevant details about the part to be loaded (both this routine and no: 115 are best POKEd into REM statements in part 1). What this routine does is read the irrelevant header (and promptly forgets it) - routine 115 then reads your program part as a headerless file.If it sounds like hard work, it is - but then it will be harder work for the person trying to copy/break in to your program, as a load other than with part 1 will cause a crash.
Combine this method with some of the other protection routines scheduled in 2.11 of Chapter II, and you will be using techniques as advanced or more advanced than those employed by the largest software houses !

118 ATTRIBUTE RESET
The CLS£ command, which works only with an Interface One attached, is not scheduled anywhere in the Spectrum manuals but performs in one stroke: PAPER 7:INK 0: BORDER 7: BRIGHT 0: FLASH 0: OVER 0: INVERSE 0: CLS (resetting all attributes to defaults)
Note that if X=INK + 8*PAPER + 64*BRIGHT + 128*FLASH then:-
POKE 23693,X sets the global ATTR value
POKE 23624,X sets the ATTR value for the lower two lines of the screen.
Note that the changes by the POKEs will become effective only when the screen is cleared.

119 LOWER SCREEN$ CLS
This routine does a CLS to the bottom part of the display only.
Incidentally, to INPUT at any position X,Y on the upper screen (INPUTs are usually only in the lower screen) use INPUT AT 22,0; AT X,Y;"Optional Message";Variable. This method works with INPUT LINE as well.

120 TRACE VARI-SPEED
When called, this non-relocatable routine will print continuously on the lower left-hand corner of the screen the line and statement number currently being executed by BASIC.
POKE 56777, Speed of execution (1=Fast through to 255=Slow)
Slow speeds are useful to make the line + statement number display readable, and also allow a fascinating insight into the ROM routines (see the example for this interrupt-driven routine). Trace remains on until you enter RANDOMIZE USR 56814 .

121 PARTIAL CLS
This routine does a CLS to the lower X lines of the upper screen (also refer to routines 119 and 122).
POKE start address + 1, X

122 LOWER UP-SCROLL
This routine scrolls up the bottom X lines of the screen a CHR$ square at a time, together with the attributes. Compare this routine with routines 3 & 121.
POKE start address + 1, X

123 ANTI-MERGE PROGRAM
Using MERGE instead of LOAD is a way of stopping most autostart programs. However programs >7K in length can be made MERGE-proof by making the first line a REM message, then entering LET X = PEEK 23635 + 256*PEEK 23636 : POKE X,60: POKE X+1,0

124 DISABLE BREAK
Use POKE 23613,PEEK 23730-5 in the autostart LINE.

125 SUPER-CATALOGUE
Call the routine with this program :-

```
1 CLS £: INPUT "Enter Microdrive";A: POKE 55993,A: CLS  :  PRINT
AT 0,0;: RANDOMIZE USR 55648: LET NO=PEEK 56072 + 256*PEEK 56073
- 6 : FOR Q=56086 TO NO STEP 11: IF PEEK Q<>13 THEN NEXT Q
2 FOR N=Q+1 TO NO: PRINT CHR$ PEEK N;: MEXT N
```

Format is Name, Type (B=Bytes, P=Prog), Length in bytes, and then either start address (if Type B) or auto line number (if Type P: 65535=> program has no auto line number).

126 REACTION TIME
LET Z=7997-USR 7997: PRINT Z/50 gives the time taken, in seconds, to hit the last key.

127 PSEUDO LOAD
To call this ROM routine, use LET L=USR 1278 (or 1248/1276/1301 /1488). A fake LOAD pattern which can be interrupted by BREAK is then displayed.

128 SEND RS232 BYTE
POKE start address + 1,Byte(followed by 2 stop bits) to be sent. If this routine is run on a just reset Spectrum, use routine 135 first.

129 RECEIVE RS232 BYTE
This routine places the byte received in location 23681. If timed out, 0 is inserted instead.
If this routine is run on a just reset Spectrum, use routine 135 first.

130 DESELECT DRIVE
This routine switches off all drives.
If this routine is run on a just reset Spectrum, use routine 135 first.

131 SELECT DRIVE
POKE start address + 1, No: (1-8) of drive to be switched on.
If this routine is run on a just reset Spectrum, use routine 135 first.

132 KEYBOARD INPUT
This routine waits for a key to be pressed, then puts its ASCII CODE into 23681.
If this routine is run on a just reset Spectrum, use routine 135 first.

133 SCREEN$ OUTPUT
POKE start address + 1, CHR$ to be sent to Stream 2 attached to Channel s (Screen$)
If this routine is used on a just reset Spectrum, use routine 135 first.

134 PRINTER OUTPUT
POKE start address + 1, CHR$ to be sent to Stream 3 attached to Channel p (ZX Printer).
If this routine is run on a just reset Spectum, use routine 135 first.

135 INTERFACE 1 INITIALISE
This routine pages in the extra 58 Interface 1 System Variables if they have not yet been created.

136 OPEN #DATA FILE
POKE 23766, Drive number (1-8)
POKE 23770, Length of Filename (1-10 characters)
POKE 23772/3, 2-byte equivalent of address of first byte of Filename
To find the address of the related Channel Area, PRINT PEEK 23670 + 256*PEEK 23671 .
Note that if the file already exists then this routine opens it for reading: if it does not already exist it is created and then opened for writing.

137 CLOSE # DATA FILE
POKE start address +2/+3, 2-byte equivalent of the channel address
Note that if the data file had been opened for writing, the current record is written to the first available microdrive sector before the file is closed.

138 ERASE MICRODRIVE FILE
POKE 23766, Drive number (1-8)
POKE 23770, Length of Filename (1-10 characters)
POKE 23772/3, 2-byte equivalent of address of first byte of Filename

139 READ NEXT DATA RECORD
POKE start address +2/+3, 2-byte equivalent of the channel address X
POKE X+25, Drive number (1-8)
POKE X+14 to X+23, CHR$ CODEs of filename with trailing CHR$0s to bring the length to 10
X+13 contains the record number which is auto-incremented. Note that it is necessary to first switch the drive motor on with routine 131.

140 SAVE NEXT DATA RECORD
POKE start address +2/+3, 2-byte equivalent of the channel address X
POKE X+25, Drive number (1-8)
POKE X+14 to X+23, CHR$ CODEs of filename with trailing CHR$0s to bring the length to 10
X+13 contains the record number which is not incremented.
POKE X+11/X+12, next data byte
Note that it is necessary to first switch the drive motor on with routine 131.

141 READ RANDOM DATA RECORD
POKE start address +2/+3, 2-byte equivalent of the channel address X
POKE X+25, Drive number (1-8)

POKE X+14 to X+23, CHR$ CODEs of filename with trailing CHR$0s to bring the length to 10
POKE X+13, Relative record number
Note that it is necessary to first switch the drive motor on with routine 131.

142 READ RANDOM DATA SECTOR
POKE start address +2/+3, 2-byte equivalent of the channel address X
POKE X+25, Drive number (1-8)
POKE X+13, Sector number
Note that it is necessary to first switch the drive motor on with routine 131.

143 READ NEXT DATA SECTOR
POKE start address +2/+3, 2-byte equivalent of the channel address X
POKE X+25, Drive number (1-8)
POKE X+14 to X+23, CHR$ CODEs of filename with trailing CHR$0s to bring the length to 10
Note that it is necessary to first switch the drive motor on with routine 131.

144 SAVE NEXT DATA SECTOR
POKE start address +2/+3, 2-byte equivalent of the channel address X
POKE X+25, Drive number (1-8)
POKE X+13, Sector number
Note that it is necessary to first switch the drive motor on with routine 131.

145 ERASE CHANNEL
This routine deallocates channel area.
POKE start address +2/+3, 2-byte equivalent of the channel address

146 CREATE CHANNEL
This routine allocates channel area.
POKE 23766, Drive number (1-8)
POKE 23770, Length of Filename (1-10 characters)
POKE 23772/3, 2-byte equivalent of address of first byte of Filename
To find the address of this Channel Area, PRINT PEEK 23670 + 256*PEEK 23671 .

147 SCREEN$ COMPRESS
This routine compresses and stores screens without attributes . To have a compressed screen stored at X, enter RANDOMIZE X & then call this routine. Afterwards, 23728/9 contains the 2-byte equivalent of the length of the compressed screen (enabling you to compute the memory saving).
Other pre-call options available are to:-
POKE start address + 21, Y : POKE start address + 47, Y
POKE start address + 22, Z : POKE start address + 48, Z
where to compress/save just the top 1/3 of the screen Y = 254 : Z = 71, for the top 2/3 of the screen Y = 252 : Z = 79 and for the whole screen but with attributes too, Y = 0 : Z = 91 .
Note that there is no loss of screen detail at all. To LOAD back a compressed screen from memory use routine 148.

148 SCREEN$ RETRIEVE
This routine retrieves any compressed screen created with routine 147. To retrieve a screen stored starting at X, just enter RANDOMIZE X and then call this routine.
The pre-call options are:-

POKE start address + 18, Y : POKE start address + 19, Z
where the values of Y and Z are as defined for routine 147.

149 OPEN # NET CHANNEL
POKE 23749, Your station number
POKE 23766, Other station number
After calling the routine the address of the net channel can be found by entering PRINT PEEK 23728 + 256*PEEK 23729 .

150 SEND # NET PACKET
POKE start address +2/+3, 2-byte equivalent of the net channel address X
POKE start address +5/+6, 2-byte equivalent of X+15
POKE X+16, number of bytes to be sent
POKE 23758, 0 for the packet to be a broadcast.

151 GET # NET PACKET
POKE start address +2/+3, 2-byte equivalent of the net channel address X
PEEK (X+13) + 256*PEEK (X+14) gives the block number of the packet requested: this is auto-incremented after each receive.

152 CLOSE # NET CHANNEL
POKE start address +2/+3, 2-byte equivalent of the net channel address X
If PEEK (X+16) > 0 at the time of calling this routine then there is still send data in the net channel - this is sent before the channel area is deallocated.