SUPERCODE

SPECTRUM

MACHINE CODE TOOL KIT

USER MANUAL

CP software

SUPERCODE
THE ULTIMATE SPECTRUM TOOLKIT

THE MANUAL

CONTENTS

by F.A. Vachha and V.B. Rumsey © 1983

Published by: CP SOFTWARE,
17 Orchard Lane, Prestwood, Great Missenden, Bucks.
HP16 0NN, England.

1. INTRODUCTION

Supercode is a toolkit program the like of which has not been seen before. It consists of ONE HUNDRED machine code routines, accessed by a state-of-the-art BASIC program (48K only). The routines are compact, and almost all are relocatable. They can be called from BASIC or machine code, either from within your program or by direct commands.

The routines can be broadly classified under two headings — utilities and special effects.

a) **Utilities** give your programming far more flexibility than you would have imagined possible—they begin where the Spectrum ROM left off. These routines include 2 types of re-number (the larger one being able to handle GOTOs, GOSUBs, SAVE A$ LINE NN, RESTOREs, LISTs, LLISTs, etc, and highlight calculated arguments), a pair of superb ON ERROR GOTO and ON BREAK GOTO (to make your programs breakproof/ crashproof), Block Moves and Deletes, Line Moves and Deletes, a Tape—Header Reader, a selelction of program-compacting routines (REMkill, Line Contract, Number— VAL String Contract), Variables Search/ List, Character Search and Replace, a number of Scrambling Routines, an ultrafast random number generator (20 times faster than that in ROM) diagnostic routines etc.

b) **Special Effects**—you name it, it's here. Every conceivable scroll, in high or low resolution, up, down, left, right and diagonal, with or without attributes, the whole screen or part of it, with selectable wrap-around/scroll-off/ripple/ shutter controls, user-definable windows etc. It's really incredible! Also, instant changes of INK, PAPER, FLASH, BRIGHT; instant filling, saving, exchanging, inverting and overprinting of screens, "impossible" border effects en masse, four types of sound generator (you can simulate whistles, horns, bells etc.), a screen interrogator (very useful in finding what is where on the screen, for collisions in your own arcade games) and much more.

Any of these routines incorporated in your own program could make you an arcade-game programmer overnight, with the colossal speed of machine code available at your command . . . Yes, SUPERCODE could repay your investment hundreds of times over.

Clearly, Supercode is a toolkit in a class of its own. It has three times as many routines as any of it's current competitors, and incorporates many features never before available to home micro users.

Now follows a summary of the BASIC program's features. Note that the program is not present on the 16K edition of Supercode due to restrictions on space. The program is not essential to the operation of the routines: 16K owners will find sufficient details about the routines from the listing which appears further on. 48K users may wish to skip the program and load the code directly using LOAD""CODE.

The program acts just like a book—it has an index of routines, and numbered pages. Each page contains all the necessary details about one routine—its name, starting address in memory, length in bytes, function/usage, how to call it (usually, but not always, with a RANDOMIZE USR command) and Save it, and in many cases a number of optional POKEs to tailor the routine for your specific purpose, say in defining the length, width and position of a screen window you want scrolled, and the attribute value you wish to be inserted. Hence, for 48K users, there is no direct need to consult this booklet at all.

A menu of commands is available at each page. Firstly, you can actually have a demonstration of the routine working, (great fun to watch,) fully under software control. This can be repeated any number of times. Further, you can either return to that page or step to the next page in numerical sequence or move to a specified page or move to the Index—each option available at the touch of a key. Lastly, you can opt to SAVE, and automatically VERIFY, the routine on tape for use in your own programs, again fully under software control. The program is error-trapped, using its own ON-ERROR-GOTO routine, and is incredibly user friendly—it will give you hours of pleasure.

SUPERCODE is in itself a matchless compendium of Spectrum routines; but when you follow the 'tailoring' suggestions this compendium will turn into a library, with billions of different routines.

We hope that Supercode gives you as much satisfaction to use as it did for us to prepare. Happy programming!

II. USING SUPERCODE

The 16K and 48K versions are both recorded on each side of the cassette:

Side 1: 16K version, 48K version

Side 2: 48K version, 16K version

1. LOADING To load SUPERCODE,

48K Version : LOAD""

16K Version : CLEAR 24572 : LOAD""CODE

a). In the 48K version, the routines are stored starting at 57344 (E000hex) and ending just below P-RAMT. In the 16K version, they are stored from 24576 (or 6000hex), exactly 32768 bytes lower than in the 48K version, to P-RAMT. In order to protect these routines from the BASIC operating system, do the following before loading the code into the Spectrum.

(16K version) CLEAR 24572
(48K version) CLEAR 57340

Note that if you have loaded the 48K version of the code after the BASIC accessing program, the latter will automatically CLEAR 57340 for you.

2. **FREE MEMORY**, 16K users will notice they have under 1K of BASIC space available when all the routines are loaded. It is suggested that immediately after loading the code the routines required by you for immediate use in the program are:

i. relocated (see Note 4) to the top of RAM.
ii. protected by CLEARING a new RAMTOP immediately below them—
iii. SAVEd (in a block if there is more than 1 routine) from their new position.

Now you should have sufficient space for your BASIC program.

3. **SAVING**, 48K users can save and verify routines under software control. 16K users can save individual routines on tape using the command SAVE "Name" CODE STARTING ADDRESS, LENGTH. Ex: for Routine 41. SAVE "MEMORY" CODE 31429,14.

4. The BASIC access program (48K only) has the following menu options:

I: To return to the start of the Index (which can be scrolled using the ENTER key). Enter a routine number to access it's page.
D: Demonstration routine, to show you what Supercode can do. Once it has finished you are returned to the Index.
P: Printer mode: Copies the current screen to the printer, if one is connected.
E: Example option, allowing you to see the routine in operation.
X: Exit mode, allowing you then either to SAVE/VERIFY (option

S) or quit to BASIC (option Q).

R: Repeat option, reaccesses the page you last looked at.

C: Continue option, turns to the next page in sequence (or to the Index if you are on routine 100).

N: Allow you to access a routine whose number is known without returning to the Index. Entering a number which is out of range returns you to the Index.

Options X, R, C, I, P and N are available each time you access a routine. Option E is available for most routines.

a) To reaccess the BASIC program at any time, use GOTO 9001. Do no use RUN, or else you will lose the colourful screen display (stored in A$) used for demonstrations.

b) This BASIC program can be MERGEd with your program, provided the latter program is short (about 1K) and does not use line numbers above 8999 (use routine 60 on it first if it does).

5. To free the maximum possible space for your BASIC program the routines in their initial positions are located as high in memory as practicable. Consequently only the last 9 User Defined Graphics (from 'M' to 'U') can be used. This is true both on 16K and 48K machines when all the routines are loaded. If you need all 21 UDG's, relocate all routines downwards by 100 (UDG's normally start at 65368/32600).

6. If the term '2 byte equivalent' confuses you, refer to routine 61. For example, the 2 byte equivalent of 64230 is 250 (more significant) 230 (less significant) because 250 x 256 + 230 = 64230

7. **RELOCATING ROUTINES;** an important feature of the routines is their relocatability. If for some reason (probably program space for 16K users) you do not wish a routine to stay at its present position in RAM, it is very easy to shift it. This is how it is done:

Ex: The routine 11, LORES LOW2/3 SCR-LEFT (low resolution, lower 2/3 of screen, scroll left), starts at 64400 and is 25 bytes long (and hence ends at 64424). To tailor the routine POKE 64416 with either 119 (for Wrap Around) or 54 (for Scroll off).

Let us say you wanted this routine to reside at memory address

49152. That would involve shifting all the bytes downwards in memory, by 64400 - 49152=15248 bytes.

This is done by the direct BASIC command. FOR X=64400 to 64400+(25-1): POKE X-15248, PEEK X: NEXT X The routine is now relocated. The address of the 'tailoring' POKE has similarly moved downwards in memory, from 64416 to 64416-15248=49168.

So the relocated routine starts at 49152, is 25 bytes long and the new address for POKEing is 49168.

The only case when this method may fail is when the routine is relocated to a region which overlaps the old one. If it does, you must proceed as follows:

i. If the New starting address is lower than the old one (as it was in the example above), the above method will work.

ii. If the new address is higher than the old one, "reverse" the order of the FOR . . . NEXT loop. By this is meant changing "FOR X=64400 TO 64424" to
"FOR X=64424 TO 64400" STEP-1" It will now work.

ROM routines and routines indicated N/A in the address table are not relocatable.
System Variable locations have addresses in the range 23552 to 23734. Where as stated above, it is necessary to relocate addresses to be POKEd wher. ocating a routine, this should not be done when the address to be POKEd is in the range 23552 to 23734. These are system variables and no change should g
"FOR X=64400 TO 64424" to
"FOR X=64424 TO 64400" STEP-1" It will now work.
System Variable locations have addresses in the range 23552 to 23734. Where as stated above, it is necessary to relocate addresses to be POKEd when relocating a routine, this should not be done when the address to be POKEd is in the range 23552 to 23734. Remember to reset RAMTOP to a suitable location, using a CLEAR command, before loading relocated code (see instruction 10).

8. The BASIC program itself has been error-proofed using routine 65 (check this by pressing BREAK, entering 'N' when is asks scroll? and by entering routine numbers such as G7 and 7G— these giving different types of errors). To disable this routine (65), use option X (exit) and then Q (quit). Restart using GOTO 9006, rather than 9001.

9. The Routines can be analysed as follows:

A) Screen Scrolling: 1-23, 37-40, 47-53, 67

B) Screen Manipulation: 24-28, 30-36, 54, 70, 72, 74-76

C) Special Effects: 43-46, 69, 73, 77-79, 89

D) Program Compression: 64, 82, 90, 100

E) Program Protection: 58, 59, 65, 66, 80, 98, 99

F) Program Manipulation: 42, 57, 60, 71, 81, 83-86, 88, 94-97

G) General Utilities: 29, 41, 55, 56, 61-63, 71, 87, 91-93

10. RAMTOP To prevent corruption by the BASIC operating system, RAMTOP should always be set to an address lower than the starting address of any machine code. On the Spectrum this is easily done using the CLEAR command.
Suppose you have a 48K Spectrum and have saved routine 12 (Lores Scroll Right). Looking at the table of addresses, you see that the starting address is 64425. RAMTOP should be set below this, so enter as a direct command CLEAR 64424.
Now load or write your basic program, which includes in it somewhere the statement RANDOMIZE VSR 64425 (This will probable be within a FOR loop e.g. FOR I=1 to 32 : RANDOMIZE USR 64425 : NEXT I) Load routine 12 from tape using LOAD " " CODE. You are now ready to go!

III. TABLE OF ADDRESSES

ROUTINE		Starting Addresses		Length
No.	Name	16K Version	48K Version	Bytes
1.	Hires Scroll-Up	31233	64001	97
2.	Hires Scoll-Down	31330	64098	99
3.	Lores Scroll-Up	3190	3190	Rom
4.	Hires Scroll-Right	31443	64211	32
5.	Hires Scroll-Left	31475	64243	32
6.	Lores Scroll-Left	31507	64275	25
7.	Lores Top 1/3 Scr-Left	31532	64300	25
8.	Lores Mid 1/3 Scr-Left	31557	64325	25
9.	Lores Low 1/3 Scr-Left	31582	64350	25
10.	Lores Top 2/3 Scr-Left	31607	64375	25
11.	Lores Low 2/3 Scr-Left	31632	64400	25
12.	Lores Scroll Right	31657	64425	25
13.	Lores Top 1/3 Scr-Rght	31682	64450	25
14.	Lores Mid 1/3 Scr-Rght	31707	64475	25
15.	Lores Low 1/3 Scr-Rght	31732	64500	25
16.	Lores Top 2/3 Scr-Rght	31757	64525	25
17.	Lores Low 2/3 Scr-Rght	31782	64550	25
18.	Ripple-Scroll Left	31807	64575	18
19.	Shutter-Scroll Left	31825	64593	18
20.	Ripple-Scroll Right	31843	64611	18
21.	Shutter-Scroll Right	31861	64629	18
22.	Lores L-Diag Scroll	31507	64275	25
23.	Lores R-Diag Scroll	31657	64425	25
24.	Screen Fill	32060	64828	30
25.	Screen Store	31976	64744	12
26.	Screen Overprint	31988	64756	28
27.	Screen Exchange	32016	64784	25
28.	Screen Invert	32041	64809	19
29.	Clear All	0	0	Rom
30.	Ink Change	32090	64858	25
31.	Paper Change	32115	64883	31
32.	Flash On	32146	64914	17
33.	Flash Off	32163	64931	17
34.	Bright On	32180	64948	17
35.	Bright Off	32197	64965	17
36.	Attribute Fill	32214	64982	44

III. TABLE OF ADDRESSES (cont.)

ROUTINE		Starting Addresses		Length
No.	Name	16K Version	48K Version	Bytes
37.	Attribute Scr-Up	32258	65026	55
38.	Attribute Scr-Down	32313	65081	62
39.	Attribute Scr-Left	32436	65204	52
40.	Attribute Scr-Right	32375	65143	61
41.	Memory Available	31429	64197	14
42.	Line Renumber	31938	64706	38
43.	Uni-Note Sound-Gen	31879	64647	28
44.	Dual-Note Sound-Gen	31907	64675	31
45.	Uni-Beep Simulator	30232	63000	10
46.	Multi-Beep Simulator	30242	63010	24
47.	Oblique Scroll-Off	30266	63034	17
48.	All-Left Scroll	30283	63051	76
49.	All-Right Scroll	30359	63127	85
50.	Hires NW-Diag Scroll	30444	63212	128
51.	Hires NE-Diag Scroll	30572	63340	128
52.	Hires SE-Diag Scroll	30700	63468	130
53.	Hires SW-Diag Scroll	30830	63598	130
54.	Screen-Print	30960	63728	49
55.	Random Number Generator	31009	63777	18
56.	Block Memory Insert	31027	63795	11
57.	Block Line Delete	31038	63806	96
58.	Chr$ Swop	31134	63902	43
59.	Chr$ Scramble	Rom Routine		**
60.	Super-Renumber	26526	59294	681
61.	Two Byte Converter	Rom Routine		**
62.	Dec—Hex Converter	27827	60595	118
63.	Hex—Dec Converter	27945	60713	113
64.	Remkill Condenser	27726	60494	101
65.	On Error Goto	28058	60826	73
66.	On Break Goto	28131	60899	72
67.	Free-Scroller	Rom	Routine	**
68.	Non-Deletable Lines	Rom	Routine	**
69.	Border Effects	N/A	60000	39
70.	Screen Search	27271	60039	123
71.	Variables Search/List	27454	60222	185
72.	24-Line Printing	Rom	Routine	**

73.	Star/Torus Draw	Rom	Routine	**
74.	Flash Switch	27394	60162	30
75.	Bright Switch	27424	60192	30
76.	Paint Fill	N/A	59136	158
77.	Record Sound	N/A	65290	28
78.	Replay Sound	N/A	65318	32
79.	Sci-Fi Character Set	24576	57344	768
80.	Protect Program	Rom	Routine	**
81.	Block Line Copy	28232	61000	400
82.	Contract Program	28632	61400	687
83.	Expand Program	29319	62087	317
84.	Expand Rem	26124	58892	244
85.	Append Statement	27639	60407	86
86.	Analyse Program	29636	62404	129
87.	Tape Header Reader	29765	62533	286
88.	Line Address	27207	59975	13
89.	Screen Grid	30051	62819	38
90.	Monochrome Program	30175	62943	54
91.	Analyse Memory	30089	62857	86
92.	Hex Loader	32582	65350	112
93.	Await Key Press	28204	60972	24
94.	Upper Case Strings	26065	58833	59
95.	Lower Case Strings	26006	58774	59
96.	Upper Case Program	25947	58715	59
97.	Lower Case Program	25888	58656	59
98.	Confuse Listing	25495	58263	135
99.	Unconfuse Listing	25630	58398	173
100.	Compress Numbers	25347	58115	148

Note: Routines marked N/A or Rom are not relocatable.

IV. DETAILS OF ROUTINES

Note: All routines are called by RANDOMIZE USR Starting Address, (the latter being found from the table given above) except where stated otherwise (eg41, called by PRINT USR Starting Address)

1. HIGH RESOLUTION SCROLL-UP
2. HIGH RESOLUTION SCROLL-DOWN

   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 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. LOW RESOLUTION SCROLL-UP

   This routine is in ROM so RANDOMIZE USR 3190 works with both 16K and 48K Spectrums.
4. HIGH RESOLUTION SCROLL-RIGHT
5. HIGH RESOLUTION SCROLL-LEFT

   Used for scrolling left or right one pixel. Use as routines 1 and 2. POKE START ADDRESS + 13, WITH 55 (SCROLL OFF), 63 (WRAPAROUND) OR ∅ (INVERSE SCROLL)
6. LOW RESOLUTION SCROLL-LEFT
7. LOW RESOLUTION TOP 1/3 SCROLL-LEFT
8. LOW RESOLUTION MID 1/3 SCROLL-LEFT
9. LOW RESOLUTION LOW 1/3 SCROLL-LEFT
10. LOW RESOLUTION TOP 2/3 SCROLL-LEFT
11. LOW RESOLUTION LOW 2/3 SCROLL-LEFT
12. LOW RESOLUTION SCROLL RIGHT
13. LOW RESOLUTION TOP 1/3 SCROLL-RIGHT
14. LOW RESOLUTION MID 1/3 SCROLL-RIGHT
15. LOW RESOLUTION LOW 1/3 SCROLL-RIGHT
16. LOW RESOLUTION TOP 2/3 SCROLL-RIGHT
17. LOW RESOLUTION LOW 2/3 SCROLL-RIGHT

   These routines will Scroll the screen one character square in each direction, leaving the attributes unchanged. Use repeated calls to the address specified in the table to scroll as far as required. To scroll attributes call first the routine above, and then one of Nos. 37-40, after defining an appropriate box and setting the attribute value to 63.

For a wrap-around scroll, first POKE the start address +16 with 119. To scroll off, POKE start address +16 with 54. (Do not POKE other numbers), eg, for routine 17 on a 16K machine, enter POKE 31798, 119. Then "RANDOMIZE USR 31782". The bottom 2/3 will then scroll right one character square with wrap-around.

18. RIPPLE-SCROLL LEFT
19. SHUTTER-SCROLL LEFT
20. RIPPLE-SCROLL RIGHT
21. SHUTTER-SCROLL RIGHT

These four routines are all pixel scrolls affecting the screen but not the attributes.

22. LOW RESOLUTION LEFT DIAGONAL SCROLL

A combination routine. First RANDOMIZE USR START ADDRESS Then RANDOMIZE USR 3190

23. LOW RESOLUTION RIGHT DIAGONAL SCROLL

A combination routine. First RANDOMIZE USR ADDRESS Then RANDOMIZE USR 3190

24. SCREEN FILL

Will fill a box on the screen with any character code.
POKE START ADDRESS + 1, WITH CHARACTER CODE'
POKE START ADDRESS + 3, WITH BOX HEIGHT
POKE START ADDRESS + 6, WITH BOX WIDTH
POKE START ADDRESS + 4, AND START ADDRESS + 7, with the "PRINT AT" co-ordinates for the top left-hand corner of the box.

25. SCREEN STORE—saves a screen in memory
26. SCREEN OVERPRINT—swaps a screen with a stored screen
27. SCREEN EXCHANGE—erases the existing screen and prints the stored one.

The three related routines all use a screen stored above RAMTOP, using 6912 bytes. 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, WITH X-256 * INT (X/256); and
POKE START ADDRESS + 2, WITH INT (X/256)

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. CLEAR ALL

This routine is in ROM, use RANDOMIZE USR ∅ with both 16K and 48K machines to simulate a power off. It not only does "NEW" but also clears RAMTOP to its original value, rests UDG's, etc.

30. INK CHANGE

    Instantly changes the ink over the whole screen.
    POKE START ADDRESS + 1 with the 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 aie called by RANDOMIZE USR START ADDRESS.

36. ATTRIBUTE FILL
37. ATTRIBUTE SCROLL-UP
38. ATTRIBUTE SCROLL-DOWN
39. ATTRIBUTE SCROLL-LEFT
40. ATTRIBUTE SCROLL-RIGHT

    For each of these a box can be defined within which the attributes will scroll. The POKE addresses are:-
    POKE START ADDRESS + 1, NEW ATTRIBUTE VALUE
    POKE START ADDRESS + 4, POKE START ADDRESS + 3 with the print at co-ordinates of the top left hand corner of the box.
    POKE START ADDRESS + 6, WITH BOX WIDTH
    POKE START ADDRESS + 7, WITH BOX HEIGHT
    For this, POKE START ADDRESS + 36, WITH 26. To cancel, POKE START ADDRESS + 36, WITH Ø instead.
    On routine 40 wrap-around can be achieved by POKEing the START ADDRESS + 43 with 26.

41. MEMORY AVAILABLE

    Called by PRINT USR START ADDRESS. This prints out the free memory available for BASIC (the area from the top of the variables area to RAMTOP).

42. LINE RENUMBER

    A short routine for use where memory is scarce. It will not renumber GOTOs and GO SUBs etc. The first line is to renumbered (L) and the interval between lines (I) can be changed by POKEing as follows:
    POKE START ADDRESS + 5, WITH I-256 * INT(I/256)
    POKE START ADDRESS + 6, WITH INT(I/256)
    POKE START ADDRESS + 8, WITH L-256 * INT (L/256)
    POKE START ADDRESS + 9, WITH INT(L/256)

43. UNI-NOTE SOUND-GEN—a programmable whistle.
POKE START ADDRESS + 1, WITH FREQUENCY
POKE START ADDRESS + 2, WITH SPAN
POKE START ADDRESS + 4, WITH DURATION
POKE START ADDRESS +23, WITH 28 FOR UP OR 29 FOR DOWN
An additional feature on routine 39 is wrap-around.

44. DUAL-NOTE SOUND-GEN—gives 2 sound channels.
POKE START ADDRESS + 7, WITH DURATION
POKE START ADDRESS + 18, WITH FIRST FREQUENCY
POKE START ADDRESS + 27, WITH SECOND FREQUENCY

45. UNI-BEEP SIMULATOR—a change from the ROM produced BEEP.
A laser zap is also available at 63950.
POKE START ADDRESS + 1, POKE START ADDRESS + 2, WITH THE PITCH
POKE START ADDRESS + 4, POKE START ADDRESS + 5, WITH THE DURATION (63951 for the laser)

46. MULTI-BEEP SIMULATOR—this can give amazing effects.
POKE START ADDRESS + 1, WITH PITCH DECREMENT
POKE START ADDRESS + 2, WITH NUMBER OF NOTES
POKE START ADDRESS + 4 AND START ADDRESS + 5 WITH THE PITCH
POKE START ADDRESS + 7 AND START ADDRESS + 8 WITH THE DURATION IN MS.

47. OBLIQUE SCROLL-OFF
48. ALL-LEFT SCROLL
49. ALL-RIGHT SCROLL
50. HIGH RESOLUTION NW-DIAGONAL SCROLL
51. HIGH RESOLUTION NE-DIAGONAL SCROLL
52. HIGH RESOLUTION SE-DIAGONAL SCROLL
53. HIGH RESOLUTION SW-DIAGONAL SCROLL
More scrolls. Use as for routines 1-23 (47 is quite spectacular)

54. SCREEN PRINT
Will print a character at any point on the screen, with any attributes (not necessarily those preset by INK, PAPER etc). The necessary POKE addresses are:

INK	START ADDRESS + 4; THIS IS PRESET AT 1
PAPER	START ADDRESS + 10; THIS IS PRESET AT 6
FLASH	START ADDRESS + 16; THIS IS PRESET AT 1
BRIGHT	START ADDRESS + 22; THIS IS PRESET AT 1
INVERSE	START ADDRESS + 28; THIS IS PRESET AT 0
OVER	START ADDRESS + 34; THIS IS PRESET AT 0

AT START ADDRESS + 40 AND START ADDRESS + 43; THIS IS PRESET AT 12, 10
CHR$ START ADDRESS + 46; THIS IS PRESET AT 42

55. RANDOM NUMBER GENERATOR

To call this routine, use let L=USR start address. A random number between Ø and 65536 will be placed in the system variable SEED. It can be retrieved by print PEEK 23670 x 256+ PEEK 23671.

56. BLOCK MEMORY INSERT

This routine inserts a given number into a block of memory. In the example used in the access program, the number 200 is inserted into the screen display. Addressese are:
POKE START ADDRESS + 1, WITH NO. OF BYTES
POKE START ADDRESS + 3 & START ADDRESS + 4, WITH THE ADDRESS FOR CODE TO BE INSERTED
POKE START ADDRESS + 6, WITH THE CODE TO BE INSERTED

57. BLOCK LINE DELETE

Lines of the program can be deleted with this code. Insert the two-byte equivalent of the number of the first line to be deleted into addresses 23728 and 23729. Then enter RANDOMIZE (number of last line to be deleted), followed by RANDOMIZE USR START ADDRESS.

58. CHR$ SWAP

This routine swops all characters of a given code with all characters of a 2nd given code within the Basic Program.
POKE START ADDRESS + 1, WITH THE OLD CODE, AND START ADDRESS + 3 WITH THE NEW CODE.
For a list of the character codes see pages 183 to 188 of the Spectrum Basic Handbook.

59. CHR$ SCRAMBLE

By POKEing 23606 & 23607 with different numbers, a corrupted character set is obtained. This can be used to protect your Basic Program. To normalise, POKE 23606, 0: POKE 23607, 60.

60. SUPER RENUMBER

Will renumber all GOTO, GOSUB, LIST, LLIST, RESTORE & SAVE . . . LINE statements. A list is produced of all calculated GOTOs, GOSUBs etc. displaying their line and statement numbers. To change the interval between lines, POKE START ADDRESS + 286 WITH THE NEW VALUE (up to 255). To change the first line number, POKE START ADDRESS + 288, AND START ADDRESS + 289 WITH THE NEW NUMBER's two-byte equivalent. Both values are initially set to 10.
If renumbering would result in line numbers greater than 9999 the interval and line number are both automatically set to 1.

61. 2 BYTE CONVERTER

Routine in ROM. This routine permits instant conversion of a number (X) from Ø to 65535 to its 2-byte equivalent. To convert X to its 2-byte equivalent, enter 'RANDOMIZE X' as a direct command. PEEK 23670 and PEEK 23671 to find the least significant and most significant byte values respectively.
For example, the 2 byte equivalent of 54321 is 212 (most sig.) and 49 (least sig.) because 212 x 256 + 49 = 54321.

62. DEC-HEX CONVERTER

63. HEX-DEC CONVERTER

These two routines auto-repeat. Enter 'Q' to return to Basic. Integers between Ø and 65535 (HEX Ø to FFFF) only are allowed.

64. REMKILL CONDENSER

Shortens and speeds up your program by deleting all REM statements in it. Use this in conjunction with no.41 to determine the number of bytes saved.

65. ON ERROR GOTO

Call this at the start of your program, say by having as line 1, "1 RANDOMIZE USR START ADDRESS". When your program is run any error will not stop the program, but will cause a jump to the program line number 9495. Here you can have any error routine you fancy. To change the error line number from 9495, POKE START ADDRESS + 52 AND POKE START ADDRESS + 53 with the two-byte equivalent of the new line number to be jumped to.

To find which error has occurred, peek 23681.
Error codes Ø, 8 and 9 are not trapped ie. they act in the normal manner. This is because they are not really errors. Note that after any error the machine stack is reset so that "RETURN" will not work.

66. ON BREAK GOTO

Similar to number 65, but only covering errors D(BREAK), H(STOP IN INPUT), and L(BREAK INTO PROGRAM). To alter the error line number, POKE START ADDRESS + 53 AND START ADDRESS + 54 with the 2-byte equivalent of the required line number.

67. FREE-SCROLLER

Routine in ROM. To scroll a screen greater than 22 lines long automatically, include the statement 'POKE 23692, 255' in your program. To scroll a specified number of lines only, POKE 23692, n where n is the number of lines to be scrolled.

68. NON-DELETABLE LINES

Routine in ROM. In order to make the first line of your program difficult to delete, POKE 23755, Ø: POKE 23756, Ø.

69. BORDER EFFECTS—Note this routine is non relocatable

Produces a colourful border effect. To customise, POKE:
a). 60006, WITH THE DURATION (From 1 to 127)
b). 60020, WITH THE COLOUR (From Ø to 7)
c). 60029, WITH THE SPACE BETWEEN LINES (From 1 to 255).

70. SCREEN SEARCH

Will find the character code at the position last printed at.
Enter: PRINT AT 7, 13; : LET L = USR START ADDRESS. Now L is the required code.

71. VARIABLES LIST

To display all variables used in your program, enter
"PRINT; : RANDOMIZE USR START ADDRESS." Useful with routines 41 and 86.

72. 24-LINE PRINTING

To print lists or text using all 24 lines on the screen, include POKE

23659, Ø before each print instruction. At the end, POKE 23659, 2, and then use 'PAUSE Ø' to prevent the scroll command corrupting the screen. Alternatively PRINT#0 and PRINT#1 commands can also be used—but experiment first.

73. STAR/TORUS DRAW

Plot a mid-screen point, then use DRAW x,y,n where x and y lie between -60 and +60, and n is in the vicinity of 8K-10K. You will be surprised—do experiment.

74. FLASH SWITCH

This routine sets every flashing square on the screen to steady and every steady square to flashing. Contrast with routines 32/33.

75. BRIGHT SWITCH

As for 74, but with BRIGHT. Contrast with routines 34/35.

76. PAINT FILL—Note this routine is non relocatable

Draw a closed convex figure on the screen (the simplest example of this is a CIRCLE). Plot a point within it, and POKE 59293 with the attribute value to be used for filling. Call using RANDOMIZE USR START ADDRESS, and the area inside will be filled; provided it is not too large (experiment with it).

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 which you can amplify the signal. Experiment with levels to optimise sound quality, but expect no miracles. Routine 77 will overwrite everything from 32768 to immediately below itself.

79. SCI-FI CHARACTER SET

POKE 23606/23607 with the less/more significant bytes of the starting address of the set (which occupies 768 bytes) less 256. You will be amazed at the change—POKE 23606, 0: POKE 23607, 60 to return to normal. As the code starts at 57344 (48K)/24576 (16K), the required POKE is 48K: POKE 23607, 223
16K: POKE 23607, 95

80. PROTECT PROGRAM

1). Introduce as line 1 a REM statement and then POKE 23755, 100. The program will work but will not LIST (until you POKE 23755, Ø).
2). POKE 23636, 150 to make the program apparently vanish. (POKE 23636, 92 to make it appear again).
3). Use INPUT LINE instead of INPUT — though you cannot now erase quotes and enter STOP, CAPS SHIFT and 6 will cause a BREAK.
4). Use routines 59 (CHR$ SCRAMBLE), 65/66 (ON ERROR/BREAK GOTO) and 98 (CONFUSE).
5). Embed colour control characters (Chapter 16 of the manual) immediately after the line number for the first line(s) of the program. Set INK and PAPER to the same colour. Then change the line numbers to Ø (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 the line no.Z read as follows:
LET ERR=256 * PEEK 23614+PEEK 23613 : POKE ERR, Ø : POKE ERR+1, Ø.
The program will now crash if you try either to BREAK into it, enter STOP for an INPUT or press CAPS SHIFT and 6 for an INPUT.

81. BLOCK LINE COPY

A block of program lines can be copied to any other part of the program. Simply "RANDOMIZE USR START ADDRESS" and follow the prompts. The new lines will be given line number Ø, and so the program should be renumbered to correct this, using

Routine 60. Note that any GOTOs and GOSUBs etc. in the new section of code will retain their original values. Any attempt to copy a block to within itself, or to a line number greater than 9999, will result in error B, integer out of range.

82. CONTRACT PROGRAM

Using this machine code routine results in the program being contracted into as few program lines as possible. This results in a saving in space, and also allows the Basic Program to run faster. The logic of the program is retained.

83. EXPAND PROGRAM

The opposite of 'CONTRACT'. You are prompted for a line number. The line whose number you enter will be expanded to as many lines as possible, to enable easy editing. The new lines will be given a line number of Ø, so the program must be renumbered afterwards to correct this. To expand the whole program, press "ENTER" on being prompted for a line number.

84. EXPAND REM

Any REM statement can be expanded by up to an extra 9999 bytes. Just follow the prompts.

85. APPEND STATEMENT

Allows you to add extra statements to the end of a program line without moving the cursor through the line. Fist move the edit cursor to the appropriate line as usual, then Enter "RANDOMIZE USR START ADDRESS". This will have the same effect as using the edit key, except that the cursor will be at the end of the line.

86. ANALYSE PROGRAM

RANDOMIZE USR START ADDRESS to find how many lines and how many statements are in your program. Useful with Routines 41 and 71.

87. TAPE HEADER READER

Call the routine, then start your tape as though to load a program. A full print of the header information will be given.

88. LINE ADDRESS

Move the edit cursor to any program line. Enter "PRINT USR START ADDRESS" to find the address of the first character in the line.

89. SCREEN GRID

After this program has been called, the character squares on the screen will become alternately bright and dark.
This checkerboard effect can be a help with 'PRINT AT' or 'PLOT' co-ordinates when designing a screen display.

90. MONOCHROME PROGRAM

Removes all colours, FLASH, BRIGHT, OVER and INVERSE, from the program listing, (thus saving space) unless they are within strings.

91. ANALYSE MEMORY

Before using RANDOMISE USR POKE START ADDRESS + 9/ + 10 with the less/more significant bytes of the address from which memory locations ar to be analysed. The columns produced are address, contents in decimal and hex and CHR$ value where printable.

92. HEX LOADER

This routine can be used only when Routine 63 HEX-DEC CONVERTER is also in memory. POKE START ADDRESS + 10 and + 11 with the 2 byte equivalent of the start address of Routine 63. POKE the system variable DEFADD (23563, 23564) with the less and more significant bytes of the address of the first byte of memory you want to load hex into. Now enter RANDOMIZE USR START ADDRESS, and enter directly in hex (Ø to FF hex). If you enter more than two digits only the last two are evaluated. Press Q to quit.

93. AWAIT KEYPRESS

Use in your programs eg. "30 LET X = USR START ADDRESS". This routine waits for a key (whose code will now be put in X) to be pressed then continues with the key code in X.

94. UPPER CASE STRINGS

Will look through the program for strings and ensure they are all in upper case characters. Watch our for INKEY$ prompts.

95. LOWER CASE STRINGS

As for 94, but changing to lower case characters.

96. UPPER CASE PROGRAM

Ensures that all program lines (except strings) are in upper case.

97. LOWER CASE PROGRAM

The opposite of no. 96. However, keywords (PRINT, TAB, LET etc) are still in upper case.

For routines 94 to 97, note that POKE 23658, Ø (lower)/8 (upper) allows you to shift the case from within the program itself.

98. CONFUSE LISTING

Changes all numbers (other than those in REMs or strings) in the program listing to a randomm code to confuse the listing. The program will still work properly, unless you attempt to edit/ change a line containing a number — in this case irreversible corruption will result. This is very useful for program protection—include a copyright REM message at the end of a line containing many important numbers. If someone attempts to delete or change your copyright message, they will render the program unusable.

99. UNCONFUSE LISTING

Undoes the effect of no. 98. The listing will now appear as normal, except for any lines irreversibly corrupted, (see above.)

100. COMPRESS NUMBER

Will remove all unnecessary spaces from the program listing by replacing any number X with 'VAL "X",' except Ø which is replaced by "NOT PI". The program will still run normally. This saves a lot of space (use no.41 to check this) but also slows down the program.