Sinclair Scene

by Sid Martin and Timothy Green

Get serious with your Sinclair – the days of Jet Set Willy are well behind us

SINCLAIR Scene reader Bill Gwillam of Swansea has written in to ask whether there are are any good spreadsheet calculating packages available for his Spectrum Plus Two.

The classic Spectrum spreadsheet was *Omnicalc 2*, launched by Microsphere in 1983. Copies are getting harder to track down, but Transform, (01) 658 6350, still sells the original version, which will work in 48K mode on any Spectrum and supports files on cassettes or the Opus, Disciple or Plus D disk systems. The program costs £14.95 – the same price as five years ago.

Microsnips, (051) 691 2008, has got the latest version of *Omnicalc*, for the Amstrad Spectrum Plus Three, on tiddlydisk at £17.95. The company also stocks the cheaper but rather limited *Mini Office* package, which costs just £5.95 and includes a simple spreadsheet as well as a word processor, database and graphics package. *Omnicalc* is probably worth the extra, if you can afford it, but at £5.95 *Mini Office* might be worth a look if you've got a fairly small and simple application.

Cool cat

We've got our first Spectrum listing this month, in the shape of *Supercat*, a machine code utility for users of the Amstrad Spectrum Plus Three.

Supercat is a souped up version of the Cat command built into Plus Three. The normal Cat command is documented in part 20 of the Plus Three manual. It prints details of files on disk or cassette. Disk contents are listed in alphabetical order, with a rough indication of the size of each file and the amount of free space on the disk.

The standard Cat command has extra features for finding the details of cassette files – the so-called 'tape catalog'. This gives you extra information which is particularly useful when your converting tape files to run from disk – it tells you the exact length of files, the address Code files were saved from and the starting line – number of BASIC files.

Supercat lets you extract the same information, and more besides, from any Plus Three disk. It will send it's summary to the screen or to the currently selected printer port. *Supercat* has been written by Nigel Mercier, the Plus Three disk expert who wrote *ZipZap*, reviewed this month.

```
10 CLEAR 32767
20 LET c=0
30 FOR i=32768 TO 33227
40   READ t: LET c=c+t
50   POKE i,t
60   NEXT i
70 READ ch
80 IF ch<>c THEN  PRINT "Error in DATA": STOP
90  PRINT "DATA loaded @ 32768": STOP
100 DATA 62,2,205,1,22,6,24,205,68,14
110 DATA 1,33,24,205,217,13,175,50,203,129
120 DATA 61,50,200,129,205,102,129,205,171,129
130 DATA 205,117,1,48,51,205,102,129,221,126
140 DATA 13,50,202,129,50,201,129,33,204,129
150 DATA 205,114,129,48,31,205,65,129,33,204
160 DATA 131,205,114,129,48,20,205,65,129,33
170 DATA 204,133,205,114,129,48,9,205,65,129
180 DATA 33,204,135,205,114,129,48,120,205,84
190 DATA 31,48,115,58,200,129,60,50,200,129
200 DATA 254,64,40,111,111,38,0,41,41,41
210 DATA 41,41,1,204,129,9,229,221,225,126
220 DATA 230,240,221,182,12,221,182,14,32,214
230 DATA 6,8,35,126,230,127,215,16,249,62
240 DATA 46,215,6,3,35,126,230,127,215,16
250 DATA 249,221,126,16,205,73,129,33,204,137
260 DATA 229,205,114,129,225,48,41,17,192,129
270 DATA 6,8,26,190,19,35,32,25,16,248
280 DATA 62,32,215,33,219,137,17,188,129,78
290 DATA 6,0,235,9,126,235,215,229,221,225
300 DATA 205,222,128,62,13,215,24,136,205,149
310 DATA 129,79,6,0,201,205,149,129,14,255
320 DATA 24,246,205,25,129,221,126,0,61,254
330 DATA 2,48,22,6,4,62,32,215,16,251
340 DATA 35,35,126,230,31,246,96,215,203,118
350 DATA 200,62,36,215,201,221,126,0,183,32
360 DATA 20,221,203,4,126,32,5,205,25,129
370 DATA 24,9,35,35,6,6,62,32,215,16
380 DATA 251,62,32,215,35,94,35,86,229,235
390 DATA 30,32,1,240,216,205,42,25,1,24
400 DATA 252,205,42,25,1,156,255,205,42,25
410 DATA 14,246,205,42,25,125,205,239,21,225
420 DATA 201,58,203,129,60,50,203,129,201,111
430 DATA 38,0,41,1,247,255,30,0,9,48
440 DATA 3,28,24,250,237,66,125,50,203,129
450 DATA 58,202,129,131,50,201,129,201,205,127
460 DATA 129,58,121,91,229,205,81,1,225,201
470 DATA 205,102,129,205,171,129,205,99,1,205
480 DATA 149,129,201,245,197,58,92,91,246,7
490 DATA 230,239,1,253,127,243,50,92,91,237
500 DATA 121,251,193,241,201,245,197,58,92,91
510 DATA 230,248,246,16,1,253,127,243,50,92
520 DATA 91,237,121,251,193,241,201,6,0,58
530 DATA 121,91,214,65,79,58,201,129,87,58
540 DATA 203,129,95,201,80,35,36,67,80,76
550 DATA 85,83,51,68,79,83,0,0,0,0
560 DATA 53333
```

Listing 1: This program loads the machine code into memory, and if you've made a typing mistake, don't worry – it'll tell you itself

```
10 REM (C) 1988 OMEGA SOFTWARE
20 CLEAR 32767
30 LOAD "SUPERCAT.COD"CODE 32768,460
40 INPUT "S=SCREEN P=PRINTER Q=QUIT "; LINE A$
50 IF A$="Q" OR A$="q" THEN  STOP
60 LET A=2: IF A$="P" OR A$="p" THEN  LET A=3
70 POKE 32768+1,A: LET X=USR 32768: IF X=255 THEN  GO TO 40
80 PRINT "+3DOS ERROR ";X: GO TO 40
```

Listing 2: Once you've got the machine code in with Lisiting 1, use this program to get the code going. The final output of Supercat can go either to the Screen or Printer.

If you're reluctant to type in all the machine code, *Supercat* is available direct from the author at Omega Software, supplied on 3" disk for £6.45. Customers who buy *ZipZap* will get a 'free' copy of *Supercat* on disk.

Cat code

The program is written in machine-code, with a short BASIC routine that traps +3DOS errors and lets you direct the catalogue to the screen or a printer. The first step in using *Supercat* is to type in the machine-code data, in Listing 1.

The first nine lines read the data values in the program, and POKE them into memory at the start of the Spectrum's second 16K page of RAM. The code is not relocatable, as it contains internal subroutine calls. Ideally it would have gone at the top of free memory, but that would clash with the workings of +3DOS, which replaces RAM page 0, which BASIC keeps between address 49152 and 65535, with page 7 containing DOS buffers.

When you run the program in Listing 1 it copies 460 bytes of machine code from DATA to address 32768 and onwards, and checks that the total of the values read is correct. If not, it prints 'error in DATA' – you should check through the DATA lines, as you've almost certainly mis-typed of them so that machine code would crash if it was called.

Once you've fixed any typing mistakes the message 'DATA loaded at 32768' will appear, and you can save the machine code with this command:

```
SAVE "A:SUPERCAT.COD" CODE
32768,460
```

You'll probably find it useful to save the BASIC loader as well, just in case you've made a devious error that has not been detected by the checking process. The last step is to type in the short control program in listing 2.

This loads the *Supercat* code and asks you to name the device you want *Supercat* to send information to. Put the disk you want to find out about in drive A, then type "S" to use the screen, "P" to use the printer, or "Q" to quit.

The appropriate channel, number, as explained in part 22 of the Plus Three manual, is stored amongst the code so that *Supercat* knows what to talk to, then the machine code is called.

The result of the call is a

+3DOS error number; normally these are one less than the error codes printed by the computer, so code 255 corresponds to '0 OK', 3 matches '4 Out of Memory' and so on.

Cat stats

Assuming all goes well, *Supercat* prints a line of information for each file. After the filename and extension comes a single character to indicate the file type : P for BASIC programs, C for Code files, dollar for string arrays and hash for numeric arrays. Then comes the exact length of the file, in bytes.

Subsequent information varies depending on the file type.

If the file contains array contents, the line ends with the name of the array. In the case of Code files, the length is followed by the address from which the code was saved – note that this information is the opposite way round from the way you normally type it in a Save command.

BASIC programs have two extra numbers besides the length.

First comes the line number where the program will start to run if you Load it. The second is the length of the BASIC part of the file – the part that contains code as opposed to variables. You can work out the size of the saved variables, in bytes, by subtracting this value from the total length of the file.

QL utility

Utilizing Software, a small QL software house, have sent in three programs for review: a SuperBASIC error trapper, command toolkit and machine code monitor.

The error trapper is the best of these offerings, and the cheapest at £7 on microdrive cartridge, with an eight-page A4 photocopied manual. Like all Utilizing Software titles, the tape is copy-protected, so you must have it in drive 2 when you load the code.

The program is just 1200 bytes long, including the protection. When you load it it grabs a further 8 bytes for each machine code procedure or function loaded, so that it can intercept the machine code from the routine. If an error occurs the utility stops the normal message, and stores the line number and error code so you can test it in your program.

The utility works in three ways. It can trap errors on commands, sending execution to the next line where you can check the error code or it can do the same for functions. These approaches tend to mean you have to add lots of extra code to your program. The third and best option sends execution to a line of your choice after an error is trapped.

The program only checks for errors in resident procedures and functions, so it can't trap structure errors, errors in expressions and syntax errors such as 'bad line'. Even so, it's useful for trapping bad names in Open and Copy, Printing to a device that is full, odd parameters for Peek and Poke word, and suchlike.

The utility works with interpreted BASIC, but not with fast compiled code. Liberation Software's *QLiberator* already includes a simple version of this technique, and a Digital Precision's *TURBO* has more than comprehensive error-trapping.

The trapper is a simple idea, but it works very well. It is clearly documented, and should suit people developing small SuperBASIC programs without a compiler, as long as they can put up with the copy protection. Utilizing Software are at 75 Oakhanger Drive, Lawrence, Weston, Bristol BS11 0RJ.

Key information

Like most things Sinclair-wise, it's possible to get used to the quirks of the QL keyboard after a little perseverance – but one little problem tends to persist. The keyboard on most production QLs has an annoying tendency to produce extra characters under some circumstances.

If you press one key then brush another as lift your finger, you often end up with two copies of the first character. This problem occurred because of a bug in the software recorded on the QL's second processor chip.

Various daft solutions have been proposed to cure this problem. It has been suggested that POKES are used to change the key repeat rate, which can't possibly help as they act on the data received by the QLs main processor, rather than the data sent from the keyboard to the 8049 second processor. Unwise people have even suggested spraying WD-40 lubricant between the keys – which eats away the delicate membrane that senses key movements!

Sinclair fixed the second processor bug, and ordered a large batch of replacement chips, but these didn't arrive until after Sinclair had sold the QL rights to Amstrad! All production QLs had the old, faulty 0.7 chip inside, but now Applied Technology, (0763) 41754 have got hold of a version 1.2, which works properly.

The upgrade is quite easy to fit, as it just involves opening up the QL and replacing the old chip with the new one. The instructions are clear and well written, and must be followed to the letter. I reckon this upgrade is £7.50 and 20 minutes well spent, if you are annoyed by extra characters appearing when you type things on a standard QL keyboard.

Wunderbar

Users of the Schön keyboard upgrade for the QL will be pleased to hear that the makers have produced their own re-programmed second processor.

Sinclair fixed the second processor bug, and ordered a large batch of replacement chips, but these didn't arrive until after Sinclair had sold the QL rights to Amstrad

The Schön Keyboard works well except that keys sometimes bounce, giving extra characters, This is not so much because of the 0.7 bug, as the keys are well spaced, but rather because of the different mechanical response of their 'proper' keys.

The £5 second processor upgrade from Schön (04865 3836) adjusts the timing of keyboard routines to avoid bounce. In every other way it's just like version 0.7. If you've got an early Schön keyboard, and keep getting unwanted characters on the screen, you need this upgrade!

Extra from space

The QL lets you plug a ROM (Read Only Memory) cartridge into the back of the computer, thereby extending the computers built in software with an extra 16K of code of your choice.

In principle the code in a ROM works just like code loaded into the computers RAM memory, but in practice there are a number of advantages if a program is in ROM. Firstly, it is available as soon as the computer is turned on, with no need to list it from disk or tape. Secondly, if a code is in ROM it does not use up any of your RAM. This advantage is crucial on a 128K QL, when you need all the RAM just to run the Psion software, or other large packages like Digital Precision's *Turbo BASIC compiler*.

Finally, as noted here last month, code in ROM runs faster than code in RAM – up to twice as quickly, in some cases. This makes ROM cartridges the ideal place to record new commands or device drivers.

In the case of the *QL Network* handler built into the SuperToolkit ROM this speed is vital – the network code can only run in ROM because it relies on precise software timings. If the code was loaded into RAM it would run at different speeds depending on the precise type of memory fitted in each machine, and different machines wouldn't be able to communicate.

The snag is that you can only plug one ROM cartridge into the QL, so you're limited to 16K of fast add-on code. Originally the cartridge port was meant to accommodate 32K of code, but half that space was later allocated to the operating system when Sinclair decided – after the launch – to incorporate a full SuperBASIC interpreter in every machine, rather than supply it as an extra on microdrive cartridge.

A few firms have come up with solutions to this problem, by providing extra sockets for ROMs that use the address space Sinclair reserved for 'hardware expansion' – disk controllers and suchlike.

In March we explained how Miracle systems garbled those 256K of spare addresses and used them to expand the QL's RAM memory up to 896K – 128K inside the computer, plus the normal 512K of expansion, plus the extra 256K. If you've got a Miracle Systems *Trump Card* you're stuck with just one ROM socket, on the back of the computer, because everything except that 16K slot has already been taken up, but if you've got a QL with 640K or less of RAM, you can use the 'spare' 256K of space to plug in extra ROMs.

RAM plus ROM

CST, (0438) 352150, sells a do-it-yourself kit of parts to adapt its RAM Plus expansion to provide four 16K ROM sockets in a space on the RAM board. The required connections are already printed on the board, but for some reason CST doesn't install the sockets or other components needed to use extra ROMs – that's left to you.

For your £10 you get a slip of paper showing where the components go, and a small plastic box containing two small integrated circuits, one transistor, four ROM sockets and a fraction of a handful of resistors and capacitors.

Soldering these in place would be a straight-forward job, if the machines used to fit the RAM chips elsewhere on the board had not filled up all the holes for the ROM parts with solder. You must suck the solder out of about 150 plated through holes before you can fit the new parts. If you've done this sort of thing before it probably won't bother you too

much, but it's not a job we'd commend to a beginner!

Once you've installed the sockets you can plug in the ROM chips from most commercial QL cartridges and use them just as if they were plugged into the normal cartridge port. we've got *Supertoolkit* 2, *Spreadscreen* and HiSoft's *'Super B' toolkit* plugged into our RAM card, so we get quite a long sequence of messages at the top of the screen when we reset the computer – one from each new ROM, plus one from the disk system.

Lost messages

These messages are read and displayed by a routine in the QL's main ROM that looks through the relevant addresses in the hope of finding and linking extra parts to the operating system. Unfortunately there's a bug in the 'AH' and 'JM' versions of the QL's own integral ROM which means that only the such part is linked, and others are ignored.

If the first part added is cleverly [illegible]n this need not be a problem, as it can look for and call the others itself. Unless you're a hacker with your own form EPROM programmer you're best advised to upgrade your system to use 'JS' or 'MG' ROMs, where the code can link as many ROMs as you provide.

A few ROMs are designed to run at one address only, and so won't work unless you plug them into the normal socket. This is true of *ICE*, Eidersoft's glitzy *Icon Control Environment* and the protection ROM supplied with Metacomco's rather unreliable *C compiler*.

Even so, the fact that popular items like *Spreadscreen* and *Supertoolkit* will happily work in any 16K slot is enough to make ROM expansion an option worth c[illegible]dering for the keen QL owner.

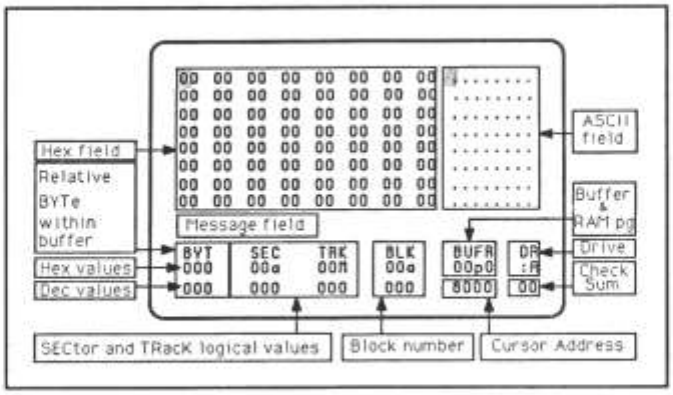
ROM alone

This information is of rather limited use unless you've got a CST RAM plus unit, or are planning to get one. We've seen advertisements for purpose made ROM add-on boards but we've yet to see the products themselves.

Italian firm SPEM offer a £34.90 unit that they say can take 3-4 (!) 64K EPROMs. Liberation Software, (01) 546 7795, offer a 'chipcard' board with a through connector and sockets for three 64K ROMs. If you've any experience of these add-ons, please write to *Computer Shopper* and let us know how you found them.

ZipZap

The Spectrum Plus Three disk system is deceptively simple to use, when all goes well, you can save or load files of any type without worrying about how they are stored on disk. Under the surface there's lots going on, and you begin to get an inkling about this when you flick through part 27 of the Plus Three manual.



The ZipZap display: This gives you a comprehensive view of what's on your disk in Hex, ASCII and any other useful information.

Most of the time you can remain blissfully unaware of the way data is stored on the disk, because the disk operating system or DOS acts as a buffer between you and all the nitty gritty details. But sometimes it's useful to be able to short circuit the operating system and look directly at the information stored on the disk.

Most disk systems support 'zap' utilities, which let you examine and edit the bytes stored on a disk. These utilities tend to be hard to use, because they require quite a lot of detailed knowledge on the part of the user, but they're very powerful. You can use a 'disk zapper' to bring back files which have been accidentally deleted, or to recover data from disks which have been physically damaged.

Zappers also let you read and write non-standard disk formats, which is useful if you want to move programs or data between different types of computer, and vital if you want to create copy-protected disks – or break the security on disks that somebody has protected!

As usual in the world of computing, anything that can be done by one computer can be undone by another. Characters known as hackers specialise in dismantling other people's software protection; luckily for us all, they're usually motivated by curiosity rather than by their greed.

ZipZap is the first disk zapping utility for Spectrum Plus Three users. It provides all the usual features for users and hackers, and lets you examine or change all data on any disk that has been created by +3DOS.

Zap pack

ZipZap is supplied on a single 3" disk, with a short A4 bound user-manual, The documentation we received was very short: just six neatly laser printed pages. Perhaps this brevity was just as well, as there was no index or contents list – there weren't even any page numbers to be found.

The documentation contains a lot of useful information, but it's disorganised and refers back to the Plus Three manual repeatedly. The *ZipZap* manual is more readable than the +3DOS documentation, but it assumes that you've read and understood the information on pages 208-219 of the Amstrad manual, and that's likely to be hard going for a beginner.

The first page is dominated by a diagram, reproduced above, which shows how *ZipZap* organises its display. The next page and a half summarise +3DOS files structure, and the rest of the manual lists the 40-odd commands, with a sentence or so of explanation about each.

You are unlikely to have much trouble if you've used a disk Zapper before, as most of the principles are the same for any floppy disk system. The *ZipZap* program has all the basic facilities you need to edit disks, and a few nice extras besides.

Omega Software is aware of the limitations of the manual, and are re – writing it at the moment. In the meantime the company has done its best to bolster the printed information with extra text in two disk files.

The 'ReadMe' file contains to the printed manual, telling you about new commands and explaining how the program can be customised to work with most types of printers

The 'Examples' file contains complete worked examples, including screen displays, showing how large and small deleted files can be reinstated in the directory, and how *ZipZap* can be used to examine or edit the contents of memory or program files on disk. These examples should be built in to the main text by the time you read this.

Disk structure

The organisation of a disk is Known as its 'format'. Normally a Plus Three disk has forty concentric rings of information recorded on it. Each ring is known as a track, and a magnetic 'read/write head' can be positioned over any track to read information thereon. The head works rather like the stylus on a record player, but it follows a circular rather than a spiral track.

The quiet clicking you hear from the drive while it works is the sound of the head moving in or out, stepping from one track to the next. *ZipZap* lets you move the head back and forth at the press of the left and right arrow keys.

Disks are divided radially as well as concentrically. Each track holds nine 'sectors' of 512 bytes, with a mark at the start of each to indicate the sector number – an arbitrary value. The disk turns through a complete circle five times a second, so the computer can find any data on the disk by just moving to the right track and waiting for the sector number it wants.

ZipZap lets you go directly to any track and sector and examine the data there, but that can be a rather hit or miss process – something like editing an encyclopedia by opening it at random and correcting any mistakes you happen to find! Normally you access the disk by looking through the 'directory' information which +3DOS keeps as files are written on the disk.

It would obviously be pretty tedious if users had to keep a record of which tracks and sectors were used, loading and saving files a sector at a time. Some early Spectrum disk systems worked just that way, but thankfully +3DOS is more sophisticated.

Indirect access

+3DOS divides the disk into 173 pairs of sectors. Each pair is called a 'block', and can hold 1K of information. Files are made up of sequence of blocks long enough to hold all the information in the file. A block can only be used by a single file, so there can be up to 1023 bytes of unused space in the last block, at the end of any file.

The disk is dived into these fairly large lumps so that a short table called the 'directory' can be used to keep track of each block, recording the file it belongs to. Unused blocks are allocated to files as they are needed. It doesn't matter where a block is located physically on the disk, as the directory lets the computer find its way from any given block in a file to the next.

The directory is always kept in two blocks near the outside of the disk. *ZipZap* lets you call the start of the directory onto the screen with a single key-press. You can then read through it or change any part of it.

The directory is divided into 32 byte sections, rather confusingly called 'Disk File Control Blocks'. Each section corresponds to a particular file, and holds the name and type of the file, as well as information about which blocks on the disk hold the file data.

The directory has a fixed size of 2K, so it can accommodate 64 sec-

If you're feeling really clever you can write new tracks with alternative formats

tions, each 32 bytes long. This means that you can only have 64 files on each disk. The limited space in each section means that if a file is more than 16K long it will have several sections in the directory allocated to it.

When a file is opened the Plus Three uses the directory as an index. It reads through the directo[illegible]looking for the name, and then uses the information in the relevant section to find the file data elsewhere on the disk.

When a file is deleted, +3DOS marks the start of its directory section or sections with an 'E5' byte. The rest of the information is retained, but the file cannot be accessed because the system sees the 'E5' and decides that the section is no longer in use.

At this point the data is still recorded on the disk – but it may be over-written if a new file is created later, as new files just use the first free space they can find. You can bring back the file by removing the 'E5' markers, but you may have lost all or part of the contents if you have written new data to the disk since the file was deleted.

[illegible]ometimes disks become physi[illegible] damaged, so that you can't read a certain point in a file. *ZipZap* lets you change the information in the directory so that the system skips over the faulty data, and you can still read the undamaged part of the file.

If you smash the directory of a disk you can still retrieve files, as long as you know enough about the data to be able to find the correct sequence of blocks on the disk. This is generally a lot easier with text files than with machine code programs!

Zap practice

In use, *ZipZap* is much like a machine code monitor. A window on the screen lets you see the contents of an area of memory called a 'page'. The *ZipZap* display, taken from the *ZipZap* manual, shows the display format. The top part of the display shows the values in the current page, in numeric and textual form, and the rest of the screen tells you what part of the disk or memory you're looking at.

Each display page consists of 128 bytes, because of the Spectrum's limited screen resolution. Internally the program works sectors of 512 bytes, so every sector is displayed in four quarters, labelled A to D. You can look through an entire sector by swapping between pages.

Commands let you read any track and sector, or the first sector of a particular numbered block. It's easy to change the contents, by typing new values as text, moving the cursor over the top of the displayed information. The cursor changes colour if you press Caps Lock.

Alternatively, data can be entered as hexademical bytes – single values between 0 and 255, expressed as pairs of alphanumeric characters.

Machine code programmers will be used to hexadecimal, but those of us with the complement of fingers and thumbs will be disappointed that values can't be entered normally, using characters 0-9. The program will convert pairs of hexadecimal characters into the corresponding decimal value on your behalf, but you have to convert the other way without help.

ZipZap is only about 5K long, and reserves all the memory above address 32768 for use as 'buffers'. This memory is divided up into 64 areas of 512 bytes. You can copy information from disk sector to one of these areas, swap areas around, or copy them back to the disk. Normally you would use the buffers when reconstructing a file assembling it sector by sector in the buffers before writing it to another disk.

ZipZap lets you return to BASIC at any time, and restart later losing without the contents of the buffers, so you can save, load or edit buffers from ZX BASIC if you don't want to do it from inside *ZipZap*. The only limitation is that there's only about 2K of free space for BASIC when you've got *ZipZap* loaded. There's no way to free buffer space to allow more room for BASIC

In essence *ZipZap* is a program to read and edit sectors, on disk or in memory, but it has a few other useful tricks up its sleeve. The 'S' key lets you scan the disk for any sequence of up to six characters, which must be entered in hexadecimal. The computer automatically searches for those bytes, reading the disk at a rate of about 1K a second and stopping whenever it finds them.

Scan gives a continuous display of the track and sector being read, and you can stop it at any time by pressing Break. The 'F' option stands for 'Find', and works much the same way but faster, as it looks through the buffers in the memory. The 'H' key displays a list of valid commands spread over three display pages, with a line of line help about each one.

Other options, aimed at high-powered users, let you select which of the Plus Three's eight 16K RAM sections appears between addresses 49152 and 65535 (C000 and FFFF for those who speak hex!), and call DOS routines using machine-code loaded into a buffer

If you're feeling really clever you can write new tracks with alternative formats. *ZipZap* lets you write tracks one at a time, using a wide range of sector numbers and sizes. You choose these by building tables in memory to tell +3DOS the format you want.

ZipZap can only cope with formats allowed by +3DOS, and although that's quite a wide range it does not include every possibility. In theory, programmers determined to discourage hackers could produce formats invisible to *ZipZap* by writing directly to the disk control hardware, but they'd need to write a lot of new code as they would not be able to use the +3DOS routines.

Verdict

ZipZap is a crude but useful utility; if you make heavy use of the Plus Three disk system it will probably pay for itself within a few months, as long as you can afford a day or two of time to learn to use it. If your a hobbyist, experimenting with the computer for fun, *ZipZap* should keep you occupied for quite a while – there's nothing else like it on the market.

Contact

ZipZap £12.95: Omega Software, PO Box 21, Shepperton,Middx TW17 8BY (0932) 228649

Please send your Sinclair news, ideas, products to be reviewed and your Club details to us here at *Computer Shopper*, 14 Rathbone Place, London W1P 1DE.