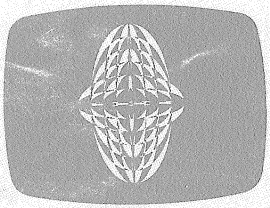
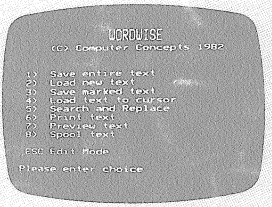
£1.00

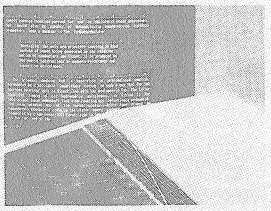
BEEBUG FOR THE BBC MICRO

Vol 2 No 2 JUNE 1983



ELLIPTO



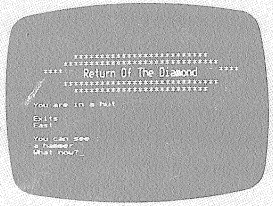


WORDWISE AND VIEW COMPARED

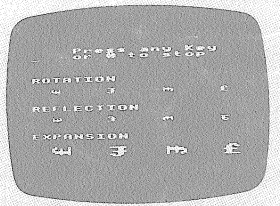
MAGAZINE CASSETTE OFFER

SOUND WIZARD

NEW FX CALLS AND NEW OS 1·2 NOTES



RETURN OF THE DIAMOND



ROTATED AND EXPANDED CHARACTERS

HEDGEHOG

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 18,000

ACORN'S SECRET MESSAGE

In an early issue of BEEBUG we mentioned that a section of the operating system ROM (in fact 3/4k at &FCØØ) is not used because the input/output devices (such as the video controller chip, the disc interface etc) are located there. We hinted that there might be some interesting data stored there. Now that the 1.2 operating system has become available, several members have checked this section of the ROM externally. This is what it reads in version Ø.1:

```
(C) 1981 Acorn Computers Limited.Thanks are extended
to the following people, companies and locations, contributors
(among others too numerous to mention) to the development
of the BBC Computer:David Allen,Bob Austin,Ram Bannerjee,Paul
Bond,Allen Boothroyd,Cambridge,Cleartone,John Coll,Computer
Laboratory,Chris Curry,Designers of the 6502,Jeremy
Dion,Tim Dobson,Joe Dunn,Paul Farrell,Ferranti,Steve
Furber,Jon Gibbons,Lawrence Hardwick,Dylan Harris,Hermann
Hauser,Hitachi,Andy Hopper,ICL,Martin Jackson,Brian
Jones,Chris Jordan,David King,David Kitson,Paul Kriwaczek,Peter
Miller,Arthur Norman,Glyn Phillips,Mike Prees,John
Radcliffe,Peter Robinson,Richard Russell,Kim Spence-Jones,Graham
Tebby,Chris Turner,Adrian Warner,Wilberforce Road,Roger
Wilson,Alan Wright.
```

The new operating system contains a different version of this message with some corrections and some altered acknowledgements. You may be interested to compare the two.

If you would like to take a look for yourself, using the Beeb, you will need a machine code monitor written entirely in machine code itself, and not making use of Basic. BEEBUGSOFT's EXMON is suitable for the job. First install the old Ø.1 ROM in one of the spare ROM sockets (leaving 1.2 in the operating system socket). Change the contents of address FE3Ø(hex) so as to select the paged ROM socket containing the Ø.1 ROM. The far left socket is selected with &FC (ie ?&FE3Ø= &FC), the next, &FD, then &FE, and &FF. Now use EXMON to examine the contents of &BCØØ to BEFF. To look at the 1.2 version, swap over the two ROMs, and repeat the exercise.

Thanks to Richard Jozefowski, I. Sommerville and M. Christiansen for the answer to the riddle.

NOTICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BO

This month's hint winners are Malcolm Inglis (£1Ø), Richard Bean (£5) and Bob Tennent (£5).

BEEBUGSOFT needs experienced 65Ø2 machine code programmers to work on programs of various kinds for the software library - generous lump-sum payments and royalties are offered. Write to: The Software Manager, BEEBUG, PO BOX 5Ø, St. Albans, giving samples of your work, and details of work in progress.

See supplement for product reviews and forthcoming events.

BEEBUG MAGAZINE

GENERAL FEATURES

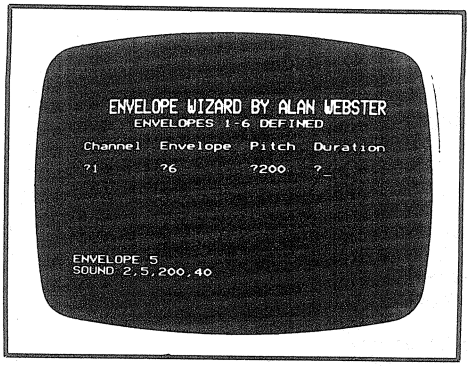
HINTS, TIPS & INFO

PROGRAMS

Program tested on O.S. 0.1 and 1.2

SOUND WIZARD (16k)

by Alan Webster



This program allows the user to produce a variety of sound effects by combining any one of the preset sound envelopes with user-chosen sound parameters. There is a facility to display the sound and envelope parameters chosen at any point so that they may be 'copied' into any program.

Type the program in and RUN it. The computer should ask you:
Which sound channel (0-3)
Which envelope number (1-n)
where n is the number of envelopes that you have defined
Choose pitch (0-255)
Choose duration (0-255)

The sound will then be heard, and the SOUND parameters displayed. You may now enter a new set of parameters, or if you press ESCAPE the computer will tell you at which program line the appropriate envelope is to be found (for you to transcribe into another program).

If you wish to add envelopes (or change them), it would be best if the number of the envelope corresponded with the line number where it is to be found, for example envelope 1 is currently at line 10, envelope 6 is at line 60 and so on. You should NOT renumber as this will confuse correspondence.

It should be noted however that only 4 envelopes can be defined on OS 0.1, whereas you can have up to 15 on OS 1.2. If you are using the old operating system, then don't type in lines 50 to 80. You may of course choose which four of the eight envelope definitions to put into lines 10 to 40.

Given below are some examples to try:

SOUND	ENVELOPE	PITCH	DURATION
1	1	200	40
1	2	200	50
1	3	4	100
0	3	5	40
1	4	50	100
1	5	200	150
1	6	0	10
1	7	20	100
0	8	6	20

```
    0 REM   SOUND WIZARD
    1 ON ERROR GOTO 200
    5 E=0
   10 ENVELOPE 1,1,-15,-15,-15,230,230,
230,30,5,0,-10,126,126
   20 ENVELOPE 2,1,4,2,-2,5,5,5,120,0,0
,-8,126,0
   30 ENVELOPE 3,1,6,0,-6,200,100,200,1
0,2,0,-1,120,110
   40 ENVELOPE 4,1,-50,-50,-50,20,-20,2
0,20,4,0,-5,120,90
   50 ENVELOPE 5,1,-100,100,-100,100,-1
00,100,20,10,0,-10,100,90
   60 ENVELOPE 6,1,10,10,10,230,230,230
,127,1,0,-1,126,126
   70 ENVELOPE 7,3,0,1,0,0,255,0,127,0,
0,-127,126,126
   80 ENVELOPE 8,3,0,0,0,0,255,0,127,0,
0,-127,126,126
   90 CLS
  100 PRINT'CHR$141;CHR$148;"  SOUND WI
ZARD BY ALAN WEBSTER"
  110 PRINTCHR$141;CHR$148;"  SOUND WIZ
ARD BY ALAN WEBSTER"
  120 PRINT"        ENVELOPES 1-8 DEFINED "
  130 PRINTTAB(0,5);" Channel  Envelope
  Pitch  Duration"
  140 INPUTTAB(1,7),C:INPUTTAB(10,7),E
  150 INPUTTAB(20,7),P:INPUTTAB(27,7),D
  160 SOUND C,E,P,D
  170 PRINTTAB(0,16);"SOUND ";C;",";E;"
,";P;",";D;SPC(100);
  180 PRINTTAB(0,7);SPC(40);TAB(0,15);"
ENVELOPE ";E;SPC(5);:VDU30
  190 GOTO 100
  200 ON ERROR OFF
  210 IF ERR<>17 REPORT:PRINT " at line
 ";ERL:END
  220 IF E=0 PRINT'':END
  230 PRINTTAB(0,20);"LIST line ";10*E
  240 END
```

USING PRINTERS

by David Graham

A printer is probably the first peripheral that most people buy for their micros after a cassette recorder and TV. Following our review of printers in issue 9, here are some notes on using printers with Beeb, explaining screen dumps, the advantages of parallel over serial printer interfaces, and generally how to get a printer up and running on the Beeb.

Most popular printers are easy to hook up to the Beeb, and can be easily controlled to produce text or program listings. The commands for this are all available directly from the keyboard. You do not need one of the many screen dumps published in this magazine - these are for producing graphics printouts, of which more later.

SERIAL OR PARALLEL

Printers can be connected to the Beeb (on the model B only) through one of two so-called 'ports' - the serial (or RS423) port, or the parallel port. Generally speaking you are better off with the latter configuration, since the Beeb assumes a parallel port unless informed otherwise. You can also print a catalogue of the contents of a cassette with a parallel printer, but not a serial one. Most popular printers can be purchased with a lead ready-made for the Beeb's parallel port, though it should be said that parallel port cables tend to be more expensive than the serial ones.

If you are using the serial port, you must execute two FX calls at some point after switching on (or resetting) the Beeb, and before attempting a printout. The first is *FX5,2. This tells the machine to route information to be printed, via the serial interface socket. The second is *FX 8,N where N is a number which determines the data transmission rate (or baud rate). Your printer manual will tell you which rate to use, and you can get the corresponding value of N from the BEEBUG reference card, page 3.

The commands for printout are the same for either printer interface (the only difference being that the parallel port does not require FX 5 and FX 8 to be used beforehand). If you type VDU 2 <return> this turns the printer on, and any text going to the screen (as the result of a PRINT statement, or the LIST command) will be printed. The printer is turned off using VDU 3 <return>. Alternatively you may use CTRL B (i.e. press CTRL and B at the same time) and CTRL C to start and stop printing. The VDU 2 and 3 calls can be used within a program - as can the FX calls, but nothing may follow an FX call in a line of Basic. Thus the following will print the word "Bananas" on the printer, and "Bananas" and "Oranges" on the screen:

```
10 VDU 2
20 PRINT "Bananas"
30 VDU 3
40 PRINT "Oranges"
```

If you want to send text to the printer only, you can use the command VDU 21 to turn off the VDU, and VDU 6 to turn it back on again afterwards. You cannot use the more obvious *FX3,3 for reasons discussed elsewhere in this issue. Thus the following program will print the word "PRINTER" on the printer only, and "SCREEN" on the screen only:

```
10 VDU 2,21
20 PRINT "PRINTER"
30 VDU 3,6
40 PRINT "SCREEN"
```

In line 10 two VDU calls (ie. VDU 2 and VDU 21) have been merged for reasons of economy.

Note: If your printer does not perform auto line-feeds, you will need to do one of two things to achieve paper advance at the end of a line during normal printing - either set the auto-line feed switch found inside many printers, or execute *FX6 before printing - this allows the Beeb to send line-feed characters. If your printer does perform auto line-feeds, then *FX6 will double-space everything. It may be cancelled by *FX6,10.

CONTROL CHARACTERS

Most printers use certain characters for control purposes. For example, if you send character 14 to an Epson printer, it will begin to print double-width characters. Character 10 will cause a line feed, and on many printers character 12 will cause a form feed. Your printer manual will give a list of these codes. You may notice that some are preceded by the word 'Escape' or ESC. This means that two characters must be sent to activate a given feature. Thus for example on an Epson, "ESC E" is given as the code for turning on emphasised character printing. This just means that the control code consists of two characters, a 27 followed by 69. 27 is the ASCII code for Escape, and 69 is ASCII E (see BEEBUG reference card, page 5).

To send control codes to a printer on the Beeb, you must first send VDU 2 to turn the printer on (not required on OS 0.1), then VDU 1 must precede EACH VDU control code. Thus to initialise emphasised character printing on an Epson requires the following.

```
10 VDU 2,1,27,1,69
20 PRINT "This is Emphasised"
```

As in the earlier example, the VDU calls have been strung together.

The user keys can conveniently be programmed for use with the printer. For example *KEY0 VDU2,1,27,1,69|M will set key 0 to give the Epson emphasised character code. Note that the character before the M in this string is a vertical slash character found on the key to the left of the left cursor key. The serial FX codes can be conveniently programmed also - but remember to put a carriage return (ie. |M) between each.
Eg: *KEY0|M*FX5,2|M*FX8,4|M*FX6|M

SCREEN DUMPS

Most, but not all, modern dot matrix printers (ie printers whose letters are made up of collections of dots rather than daisywheel types which give solid letters) may be used to print dot patterns as well as text; some daisy wheels are also capable of this. In this former mode data is sent to the printer which specifies not which letter of the alphabet to print, but which combination of dots to put on to the paper at any given instant. Using this facility it is possible to reproduce screens of graphics in modes 0, 1, 2, 4 and 5. Such a translation from dots on the screen to dots on the matrix head can be performed by a so-called screen dump program.

As its name implies, this dumps the contents of the screen on to the printer paper. If the dump is written in Basic it can take up to 20 minutes to dump a screen, but in machine code, times of better than one tenth of this are possible. See BEEBUG vol.1 no.9 for machine code dumps for the Seikosha and Epson.

REFERENCES

a) BEEBUG Vol.1 no.1 p.20 Installing a parallel printer port (but note that the Beeb PCB error mentioned here has been fixed on production machines).

b) BEEBUG Vol.1 no.9 p.8 Machine Code Dumps for Seikosha GP 100 and EPSON FT 80 & FT 100.

c) BEEBUG Vol.1 no.10 p.9 Printer Review - including Seikosha GP100 and Epson FT80 and 100.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

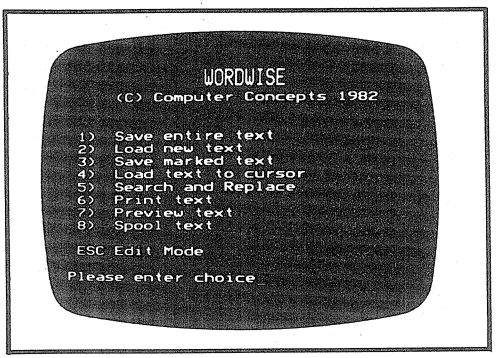
QUICK KEY CHECK

Use INKEY(-129), optional brackets, to test if ANY Key is being pressed. Like all the INKEY with negative argument statements, this tests the keyboard and not the buffer, but, unlike the 'normal' negative INKEY, which can only test a specific key, it will test all keys except CTRL and SHIFT, (including the 'soft' keys).

It is negative true logic, ie IF NOT INKEY(-129) will indicate a key IS being pressed, whilst IF INKEY(-129) will indicate if a key is NOT being pressed. Richard C. Bean.

VIEW OR WORDWISE—A COMPARATIVE REVIEW

by Mike Williams



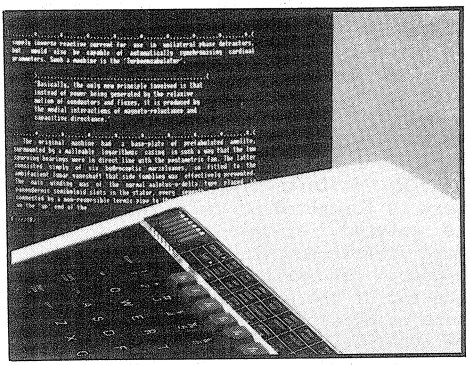
WORDWISE
Supplier: BEEBUG
Cost: £40 all inclusive
£45 with new ROM

VIEW
Supplier: Acornsoft
Cost: £59.80 inc VAT and new ROM if requested
plus £9.95 Tape to configure printer

WORDWISE is a word processing system that has been available for several months and has become very popular with BEEBUG members. Now, other rivals are emerging to challenge WORDWISE, in particular, VIEW from the Acornsoft stable. Before writing this review, I had not used either of these two systems before, although I am familiar with other word processing systems.

WORDWISE impressed me very quickly as it was easy to install the chip, the WORDWISE manual was straightforward and easy to follow and I had no difficulty in starting to enter some text. All text entry takes place in mode 7 which is easy to read but, of course, gives no feel for the final formatted appearance of the document. The cursor control keys, by themselves, and also in combination with SHIFT and CTRL keys, enable the cursor to be moved very quickly backwards and forwards through the text. Using WORDWISE for this is really very nice indeed. The usual tasks of deletion and insertion of text also proved very easy to master. There is no specific command to delete a line of text, though CTRL/A seems to cope with deleting blank lines quite well.

There is also a search command using a function key, but this only allows you to search for a single character which is not very useful. However, if you return to the main menu, option 5 allows you to search for a whole string of characters and replace each occurrence with a new character string, which is much more useful. There is also a function key which allows you to delete from the position of the cursor up to a single specified character, which again is rather restricting, and even if you repeat the process till all the appropriate text is deleted, the process does not behave in a logical way. Large areas of text are best deleted by setting markers at the start and end, and using a different function key to delete marked text. Similarly, whole sections of text can either be moved or copied to a new position in the text, all very useful and very easy.



In order to see the formatted appearance of the text, you need to return to the main menu where option 7 displays the formatted text on the screen in mode 3, an 80 column mode. All formatting is controlled by special commands which are embedded in the text of your document but not printed. All such commands are displayed in green

when editing text, which is an excellent idea. All the commands have default values, so it is only necessary to include them when you want to format differently. The range of commands is very comprehensive. However, the fact that you can only see the formatted text by returning to the main menu and selecting option 7, I find rather tedious and time consuming and the inability of WORDWISE to perform on-screen formatting, the most serious disadvantage of the system. This is particularly true when setting up tables and other critical text layouts.

Having said that, the great virtue of WORDWISE is its ability to let you get on with the job and not to stand in your way. Because of this most people will, I am sure, put up with the few disadvantages for the sake of its power and simplicity.

Like WORDWISE, VIEW is supplied as firmware - in fact VIEW is in ROM, while WORDWISE is in EPROM. Fitting was again quite easy but using the system proved more difficult initially. After some frustration and not a little heart searching I eventually discovered from the manual that VIEW is selected by typing *WORD and not as I had logically expected *VIEW. Unlike WORDWISE, you are not presented with an initial menu. Instead you need to know what commands to type in - back to the manual (there are two manuals actually). To start creating a new document requires the command NEW followed by hitting the ESCAPE key.

VIEW differs from WORDWISE in that text is formatted on the screen as it is entered, allowing you to see much more clearly how the final pages will appear. You will also see a ruler at the top of the screen. The ruler is a line of characters showing the position of the left and righthand margins and the positions of any tab stops. VIEW allows you to use up to 100 different rulers within any one document. Each ruler controls the formatting of text up to the next ruler. The top line of the screen also contains additional information on which of several options are currently selected.

In VIEW, all the editing functions are carried out by using the function keys, alone and in combination with the SHIFT and CTRL keys giving some 30 functions plus the cursor control keys as well. This avoids the use of any control keys, but the function key layout is not that obvious and takes some getting used to. In general, VIEW will perform all the functions and formatting of WORDWISE and some extras as well. VIEW also allows the screen to scroll horizontally covering lines of up to 132 characters. Splitting and joining lines is easy and also line deletion. Markers (up to 6) can be used to mark sections of text for deletion, moving or copying. Using the ruler facility, you can very easily reformat text, with or without justification. VIEW also uses the equivalent of the embedded commands used by WORDWISE but with VIEW the commands are inserted in the lefthand margin of the screen rather than in the text itself. Again the facilities are similar with several extras.

Both VIEW and WORDWISE provide a range of more advanced features. However, most word processing users tend to use only a relatively small proportion of the total facilities available, and it is from this point of view that I have tried to sum up the relative merits of these two systems. WORDWISE really is easy to use. It positively encourages you as you are using it. For many, these virtues and its lower price will be sufficient reason for purchase. It is ideally suited to the home hobbyist. For those who will need to handle more complex text layouts with documents running into many pages, the on screen formatting and use of rulers by VIEW will ultimately save much time and provide a more powerful environment for this type of work, once the initial learning difficulties are overcome. For those who are used to sophisticated word processing packages on business micros, VIEW will still leave a lot to be desired.

In conclusion, if I was buying a word processor for myself then I think my choice would be WORDWISE; if I was buying for my work, (i.e. not paying for it myself) then I know I would choose VIEW.

NOTE ON PRINTER CONFIGURATION

WORDWISE will work with any printer that works with the BBC micro. You configure WORDWISE for your printer directly using *FX commands (if required), and embedded format commands. These may be incorporated into the function keys, and loaded from tape or disc as explained in BEEBUG vol. 1 no. 10 p. 41.

VIEW needs to be configured if you want to use the special features of a given printer (e.g underline, emphasised print etc.), and a tape can be purchased at £9.95 for this purpose for the following printers:
Ricoh, Qume, Diablo, Olivetti, Epson and the NEC spinwriter.

EDITORIAL AFTERWORD

We have been using Wordwise to produce the whole of BEEBUG magazine for several months, and are extremely happy with it. Because of this we are biased in favour of Wordwise - and for this reason we commissioned the impartial comparative review presented here. We feel however that we must add one point regarding Wordwise's use of mode 7 for text entry. This mode was chosen to leave as much memory free for text as possible; but even more important, to enable it to be used with an ordinary TV (monochrome or colour) or with a colour monitor. The full on-screen formatting provided by VIEW requires one of the 80 column modes, and these necessitate the use of a monochrome monitor. Televisions (monochrome or colour) and even normal colour monitors do not give sufficiently high resolution.

SOFTWARE UPDATE SOFTWARE UPDATE SOFTWARE UPDAT

MOON LANDER

Contrary to rumours the BEEBUGSOFT version of Moonlander WILL run on a 16k machine, but you must type PAGE=&B00 <return> before loading it.

ASTROTRACKER COLOUR CHANGES

ASTROTRACKER has a facility to alter the colours used. This is achieved as follows: alter line 1190 of the header program to 1190 ?&8E=f:?&8F=b where f and b are numbers between 0 and 7 representing the colour of the foreground and background respectively. See BEEBUG reference card for details. The flashing colours have no effect.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MUSICAL KEYBOARD ?

To obtain one of the wierdest musical keyboards you are likely to find, type in ?&FE40=0 and press return. Alan Webster

LOSING LINES THROUGH FUNCTION KEYS?

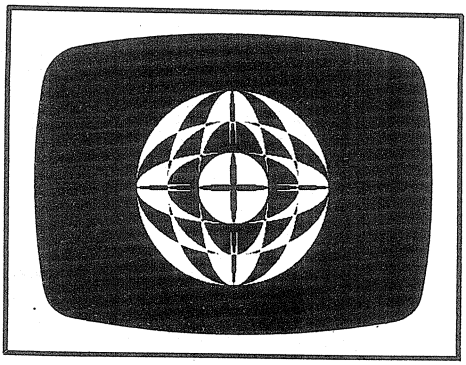
Derek Chown says that if you are not careful you can lose program lines when using function keys. You might type a number, for example, then accidentally press a function key. To eliminate the risk, arrange for the function keys to send a CTRL/U character (not VDU 21) first:

```
*K.1 |UMODE7|MLIST|M
```

Program tested on O.S. 0.1 and 1.2

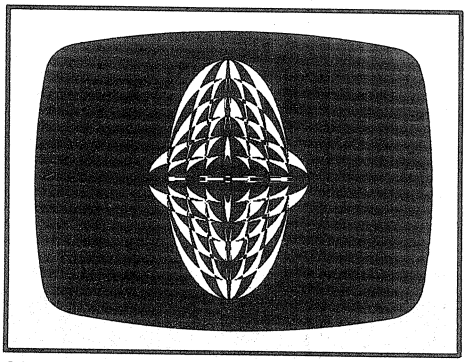
ELLIPTO (16k/32k)

by Simon Wilkinson



This is a relatively short program which produces varied patterns by plotting a succession of filled ellipses of random size. The ellipses have the same centre, and the pattern is gradually built up using so-called 'exclusive or' plotting (achieved with the GCOL 3,1 statement in line 130). This program uses mode 4 and gives patterns on a black background. It is part of a set of three similar programs, one of which produces colour displays. The full set is supplied on this month's cassette.

```
10 REM    Ellipto
20 REM An extract from S.Wilkinson's
30 REM Ellipse Plotter
40 MODE7
50 PRINT
60 REPEAT
70 INPUT"Mode 0 or 4 ",M%
80 UNTIL M%=4 OR M%=0
90 MODE M%
100 IF M%=0 M%=2
110 VDU23;8202;0;0;0;0
```



```
120 VDU29,640;512;
130 GCOL3,1
140 C%=RND(7)*16+16
150 FORA=16TO512STEPC%
160 FORB=16TO512STEPC%
170 L%=-A
180 MOVEL%,0:DRAW-L%,0
190 FOR Y%=4 TO B STEP4
200 X%=A/B*SQR(B*B-Y%*Y%)/M%:X%=X%*M%
210 MOVE X%,Y%:DRAW -X%,Y%
220 MOVE X%,-Y%:DRAW -X%,-Y%
230 NEXT Y%
240 NEXT,
250 A=GET:CLS:GOTO140
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TEXT INPUT CONTAINING COMMAS

If you want to input text that has one or more commas in it, there are two methods that you can use. The first is to use INPUTLINE, eg:

```
10 INPUTLINE A$
20 PRINT A$
```

and the second method is to enclose the text in quotes when inputting it, eg. using the short program:

```
10 INPUT A$
20 PRINT A$
```

as a test, run it and type in "A,B,C,D" with the quotes and you should see that the output from line 20 has no quotes, but does contain the commas.

It should be noticed however that the first example only works with keyboard input, whereas the second method works for any channel. C.Opie & A.Webster

ADVENTURE GAMES REVIEW

This month our software review covers adventure games. These are not real-time action games, but games where you converse in English with the computer to move around in an imaginary world and accomplish certain goals, like hunting for treasure etc. If you have never played an adventure game, it is difficult to explain how they can be so entertaining or so addictive. The fact that they ARE is underlined by the fact that Acornsoft have produced no less than three for the Beeb to date.

TITLE :Adventure Quest
BY :Level 9 PRICE :£9.95
Reviewer :J. Gregory

In this game you are an apprentice magician and are the only person who can find the four stones needed to enter the tower to defeat the Demon Lord. I found Adventure Quest confusing. I have not made a great deal of progress with it but like all good things it is addictive. There are problems mapping this game initially because when you are killed, (and resurrected), you are transported to a location which is not easy to place in relation to the previously mapped sites. However, persevere and all will become clear in the end. This game only allows you to carry a limited number of objects so you must plan carefully.

TITLE :Firienwood
BY :M P Software PRICE :£6.50
Reviewer :J. Gregory

The quest in Firienwood is to find the magic Golden Bird of Paradise which has been imprisoned in a castle in the middle of an enchanted wood. In return for its freedom the bird is said to give you health, wealth and prosperity. Being a machine code program the response time is extremely fast. An annoying feature to me, is that there seems to be a creature that appears at random and kills you without warning.

One point to remember about this game is that when you are asked if you would like another game, only press "Y" without a <return>, otherwise you will lose the program. Bearing in mind its price I consider Firienwood to be good value for money.

TITLE :Xanadu
BY :Hopesoft PRICE :£7.75
Reviewer :J. Gregory

On the face of it this is a standard "find treasure and kill dragons" adventure with a nice touch at the finish. However some of the treasures move about each time the game is played, thus making each game slightly different. There is also a facility for one or two players. The two players can be allies or in opposition and this can be changed during the course of the game. It also differs from other adventures in that there is no maximum score; your score is based on the number of moves, number of enemies killed and the treasures found.

TITLE :Sphinx Adventure
BY :Acornsoft PRICE :£9.95
Reviewer :A. Webster

Sphinx Adventure was my first real try at an adventure game and I found it totally captivating. Now, having tried more adventures, I must say that this remains my favourite. There are some awkward problems to get around, for example, getting a jack out of a pit, which took me nearly a week to figure out.

The object of the game is to collect treasures and take them to the Sphinx. During the game some nasties will appear to try to rob you of some valuable items, either that or they try to kill you.

I have nearly completed this game on several occasions but still can't find the valuable box of matches that I need!

Although the price is relatively high this game will appeal to most adventure players and even non-Adventure players like me because it is very witty in many places, for example, try killing the giant rabbit.

SUMMARY TABLE

	Save Game	Colour	No. of Locations	Speed	Price inc.VAT	Overall Value
Adventure Quest	Y	N	225	*****	£9.95	*****
Sphinx Adventure	N	N	144	***	£9.95	****
Xanadu	Y	Y	103	****	£7.75	****
Firienwood	Y	Y	60+	*****	£6.50	****

INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO

VARIABLE DATA

Incidently, has anyone else noticed that you can include variables in data statements? When read they act as if the value of the variable itself was written in its place. H.A. Fraser.

[If you would like more on this see article on 'Debugging' in vol.1 no.6 p.20. Ed]

FAULTY DATA

Nik Kelly knows of a quirk of DATA/READ/RESTORE which may have confused Beeb owners who are accustomed to other micros. Most machines will find DATA wherever you put it. The BBC (BASIC 1) is more fussy. If you tag DATA on the end of the line where it's wanted, you will either get "Out of Data", or read the DATA from somewhere else. Using RESTORE to your current line, (YES ! It WILL Renumber correctly!), will astonish you with "Out of Data at line....".

Before you despair, look closer. The Beeb is dropping the first item. To outwit it, put a dummy 0 or just a comma ahead of your DATA. eg:

```
10 RESTORE 10 :FOR N=1 TO 3 : READ M: PRINT M: NEXT N: DATA ,11,12,13
```

will happily produce 11, 12 and 13 as its output when RUN.

ARCADE GAMES HIGH SCORES

Supplier	Game	Score	Player
BEEBUG	Astro-tracker	27825	P.Miller of Essex
ACORNSOFT	Rocket Raid	54020	A.Webster of BEEBUG
ACORNSOFT	Monsters	104650	I.Cook of Essex
ACORNSOFT	Snapper	134410	I.Cook of Essex
ACORNSOFT	Arcadians	16010	I.Cook of Essex
ACORNSOFT	Super Invaders	17450	A.Webster of BEEBUG

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MULTIPLE TABS IN WORDWISE

Wondering why you can't get more than one TAB indentation at the beginning of the line by hitting TAB two or more times in succession ? The secret is to type (e.g. for a triple indentation): TAB <space> TAB <space> TAB. A.W. Rowe.

PROCEDURE FUNCTION LIBRARY

Program tested on O.S. 0.1 and 1.2

STORAGE OF MACHINE CODE WITHIN BASIC

by Brian Carroll

The useful procedure below provides for the insertion of machine code directly into an area of memory, and may be appended to any Basic program. It is a good deal more concise than using Assembler mnemonics. It is also pretty indecipherable for anyone who wants to protect his machine code.

The machine code must be placed in line 20020 of the procedure as a series of hex bytes with no separators. As an example the procedure as given contains a sequence of code to fill the screen in mode 7.

You put the procedure on to the end of the Basic program that is to be used with it, and call it using PROCp_code(addr%), where "addr%" should be replaced by a numerical address of the first byte of the block at which it is to be stored. In the example line 10 calls the procedure, which places the code into memory at address &B00, while line 20 executes it. As it stands the code will fill the screen with discontinuous white blocks. To alter the character used, change FF on line 20020. Replacing it with the value 20 will clear the screen.

One way to find out what the sequence of code should be for other machine code programs is to assemble your original machine code routine, at the place in memory where it is to run from, say &D00; and then get a print out in hex as follows:

```
FOR A=0TOn:PRINT~?(&D00+A);:NEXT
```

where n is the number of bytes of code used.

```
10 PROCp_code(&B00)
20 CALL &B00:REM *ROUTINE*
30 END
20000 DEFPROCp_code(addr%)
20010 LOCAL Z%,code$
20020  code$="A97C8571A900857018A000A
9FF917018A5706901857 0A57169008571C980
D0E960"
20030 FOR Z%=1 TO LEN(code$) STEP 2
20040 addr%?(Z%DIV2)=EVAL("&"+MID$(co
de$,Z%,2))
20050 NEXT
20060 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SINGLE KEY SAVE

B.J. Bailey has found it useful to store the current name and version of a program in a variable. He does this on the first line of each program that he writes, eg:

```
10 PROG$ = "TESTv.1"
```

The effect of this, apart from giving the name and version at the top of a listing, is that a function key can be programmed to provide a 'single-key' save. The key is programmed by:

```
*KEY 8 SAVE PROG$|M
```

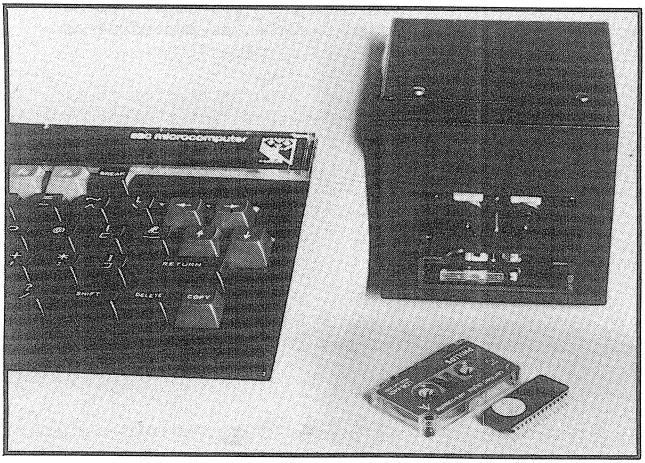
This itself could be incorporated as a line of the program (line 20, say).

Notice that using this method means that the program must be RUN before being SAVEed. [This idea uses the rather odd fact that you can save a program dynamically (ie. using a variable as a filename) when the call is made from a user key, though not when it is made from within a program. Ed]

HOBBIT FLOPPY TAPE REVIEWED

by Colin Cohen

Hobbit floppy tape system from Ikon Computer Products, Kiln Lane, Laugharne, Carmarthen, Dyfed, Wales.
Telephone (Ø99421) 515.
Cost: £155.25, second drive £138 including VAT.
Tested on O.S 1.2 with Wordwise.
It is claimed that Hobbit will also work on OS Ø.1.



The Hobbit is a so-called "floppy tape" unit which plugs directly into the Beeb providing program and data storage with some of the facilities of discs, but at a lower cost. The device actually stores programs digitally on Philips mini cassettes similar to the 3Ø-minute audio tapes used in some dictating machines. The tapes cost around £3.ØØ each, and will store about 6Øk bytes on each side.

The Hobbit plugs into the BBC user port (not present on a model A) via a twenty-way connector, while power is taken from the BBC's power out socket (not present on early Beeb power supplies), and so does not need mains voltage or a transformer. A second drive can be chained from the first and neither arrangement requires the disc interface or modification to the cassette interface. It is possible to switch from Hobbit to tape and vice versa by the commands *TAPE or *HOBBIT. Firmware is provided in the form of an EPROM. It will fit in any of the empty 28-pin sockets alongside the Basic and OS ROMs - if socket IC1ØØ is vacant you only need to slip it in and move the link marked 18 over. If this seems too complicated Ikon will do it for you for an additional £5.75. The only other chore is formatting new tapes. This takes about three and a half minutes per side following the command *FORMAT NAME and is by far the slowest Hobbit function. A 1ØØk disc takes around half a minute by contrast.

With the exception of the cassette eject button on the Hobbit the system control is from the keyboard. In other words fast forward, rewind and play/record are all looked after by the operating system. In Basic the Hobbit index is very helpful and the single command *CAT will display up to 6Ø eight-character file names (as against the ten of the cassette filing system, or the rather limited 7 of the disc system), each prefixed with a number. Data or programs can be LOADed or RUN from the index page by keying the number or name followed by f8 or f9, and sitting back.

Unwanted files can be deleted by *DELETE NAME - though data files need to be followed by: D, which is one of the many points not made in the first round of documentation. *RECOUP brings files back from the dead (if you have not already written a new file), while *KILL is by far the fastest command available and will wipe a whole tape in no time at all. You can also RENAME, COPY or APPEND (to each other) files.

The most important information not documented is the warning not to leave a tape in the drive at power on/off. If you do you may (as I did) corrupt a block which will inhibit LOADing the file back. Most BBC OS commands can be used, but not normally in abbreviated form: you have to use *CAT and not *. and you cannot interchange lower case with capitals in OS commands or file names. You can switch to Wordwise with *W. - but if you want to get back you need *BASIC, as *B. does not operate. Other undocumented items are the error messages such as CHECKSUM, HEADER, CLOSE FILES and SGN.

The read/write speed is 750 bytes/second as against the 150 of the cassette, and 11,500 of the disc. The actual speed improvement, over the cassette filing system, will depend on the sort of file that you are making: for instance there is a certain 'overhead' when it comes to making a file - the Hobbit has to find a place to write it and the fuller the tape is, the more it may have to hunt for a place to save it. Average access time is said to be 20 seconds with a maximum of 90 seconds. You must then add the time to save or load the program itself, at around one fifth of the time for loading from ordinary cassette. The disc system by contrast will save or load 20k bytes in just 5 secs. This includes access time.

The single Hobbit costs about four times more than an expensive tape recorder, which does not sound unreasonable. Comparisons with discs are perhaps less favourable. The Hobbit costs about the same as a 100k disc drive (as supplied by Microware for example - Acorn's own are more expensive), but the disc user must add the cost of the disc interface. However discs are £1 or £1.50 cheaper than the Hobbit cassettes, and this price differential will make serious inroads into the apparent relative economy of the Hobbit over a floppy disc.

BRAIN TEASER – THE 3X + 1 PROBLEM

by Gareth Suggett

This problem has been described by one author as having become part of modern mathematical folklore. I first came across it in Douglas R. Hofstadter's "Godel, Escher, Bach", but nobody seems to know its origin. The problem is very simply stated:

Start with any odd number (3, say). Multiply it by 3 and add 1 (3 x 3 + 1 = 10). Now divide by 2 repeatedly until another odd number is produced (10/2=5). Repeat the process (5x3+1=16: 16/2/2/2/2=1). Once 1 is reached the process terminates because 1x3+1=4 and then 4/2/2=1. Does every number eventually reach 1?

Nobody knows the answer, but every number so far investigated has reached 1. Some take longer than others: 3 took only 2 steps, whereas 27 takes 41 steps. What other numbers can you find with a sequence greater than 40 in length? Can you find a formula that will generate several of these "long numbers"?

If you ever tire of the many problems involved in the "3x+1" series, you may like to consider the general "px+q" problem. A particularly interesting case is: p=1093, q=1.

DISC HINTS DISC HINTS DISC HINTS DISC HINTS DISC HIN

D.T. Goodwin informs us that when using the disc operating system it is possible to open a file with no filename. Whilst you can still use this file on the disc it cannot be deleted with the DELETE command. To overcome this you can use the '*WIPE *' command selectively (a further safeguard when using WIPE would be to lock all files except the one with the blank filename).

MULTI-PROGRAMMED FUNCTION KEYS

by David Graham, K Stephenson and Tim Powys-Lybbe

The BBC micro's function keys are an extremely precious resource. So precious that 10 (or 15) keys are just not enough. In this article we show three ways to multiply the effective number of useable keys. By such methods, you can achieve 40 or more individual functions.

The three different methods presented use different techniques, and are suited to different applications or machine configurations. They can be summarised as follows:

1) DISC METHOD
   Here different key setting programs are loaded in from disc using function key 9. It can be used in immediate mode (ie. outside a program).
2) MULTIPLE STRING METHOD
   Different key settings are loaded in from a resident program using function key 9. It can be used in immediate mode (ie. outside a program).
3) MULTIPLE KEY SELECTION WITHIN A PROGRAM
   Different combinations of the function keys and <shift>, <ctrl>, and <tab> keys are detected with the INKEY command, and direct a program to different routines. This method can only be used from within a program.

DISC METHOD

The simplest version of this method uses just two sets of key definitions, and pressing f9 at any time switches between them, by calling the alternative key-set program from disc. Since this is achieved with *LOAD (rather than a Basic program) it is done quite invisibly, and only takes 1 second to complete the change.

The pair of key-sets are prepared as follows:

1) Program function keys 0-8 with various functions required for set 1. It might be advisable to do this via a Basic program so that you have a record of your settings.
2) Set key 9 as follows:
   *KEY9*L.2|M
3) Save this set as follows:
   *SAVE "1" B00+100
4) Clear the machine and repeat operation 1 for the second set of keys 0-8.
5) Set key 9 as follows:
   *KEY9*L.1|M
6) Save the set with:
   *SAVE "2" B00+100

The task is now complete, and any time you press key 9 you will alternate the key settings. To use from cold, type *L.1 <return>

You could, if you wish, use *MOTOR (or *M.) to switch the cassette motor light on and off to tell you which key-set is in use at any time, and you could increase the number to say three sets, making set 1 load set 2, set 2 load set 3, and set 3 load set 1. You could also use the INKEY command (as in the next method) to detect CTRL, SHIFT etc, and *LOAD different keysets from disc accordingly.

MULTIPLE STRING METHOD

Using this method, nine of the function keys (f0 to f8) have assigned to them three different sets of strings (eg. Basic statements). The set which will be active at any one time is selected by simultaneously pressing function key f9 and either the SHIFT, CTRL or TAB key. As you can see from line 12, each time key f9 is pressed, the program is re-run, and it checks for the SHIFT, CTRL or TAB keys in lines 14 to 18, and then loads the keys accordingly.

The following simple program should first be typed in and saved on cassette or disc with the name KEYSET:

```
10*FX15,1
12*K.9 RUN|M
14IFINKEY(-97)THEN PROCupper:GOTO 22
16IFINKEY(-2)THEN PROCmid:GOTO 22
18IFINKEY(-1)THEN PROClower:GOTO 22
20GOTO14
22END
24DEF PROCupper
26*K.0 DIM P%-1:P.HIMEM-P%|M
28*K.1 *FX
30*K.2 READ
32*K.3 DATA
34*K.4 RESTORE
36*K.5 ENVELOPE
38*K.6 CHR$
40*K.7 VDU23,
42*K.8 VDU19,
44ENDPROC
46DEF PROCmid
48*K.0 MOVE
50*K.1 DRAW
52*K.2 PLOT
54*K.3 GCOL
56*K.4 SOUND
58*K.5 INPUT
60*K.6 INKEY
62*K.7 LEFT$(
64*K.8 MODE 7|M
66ENDPROC
68DEF PROClower
70*K.0 DEF
72*K.1 PROC
74*K.2 REPEAT
76*K.3 UNTIL
78*K.4 COLOUR
80*K.5 FOR
82*K.6 NEXT
84*K.7 TAB(
86*K.8 RND(
88ENDPROC
```

To use this key definition system during the development of a program you first CHAIN it in, eg:

CHAIN "KEYSET"

and then hit either the TAB, CTRL or SHIFT key once the routine is known to be running. With the KEYSET program supplied the keys will be programmed as follows:

Set:TAB		CTRL	SHIFT
f0	memory	MOVE	DEF
f1	*FX	DRAW	PROC
f2	READ	PLOT	REPEAT
f3	DATA	GCOL	UNTIL
f4	RESTORE	SOUND	COLOUR
f5	ENVELOPE	INPUT	FOR
f6	CHR$	INKEY	NEXT
f7	VDU23,	LEFT$(	TAB(
f8	VDU19,	MODE7|M	RND(
f9	reset	reset	reset

So, if your first key strike after CHAINing the KEYSET program was SHIFT, the function keys would be programmed with DEF,PROC,REPEAT etc.

Once this stage has been reached you can start to develop your new program. It is important that you begin the program starting from at least line 100 because the KEYSET routine takes up lines before this. Each time you need one of the alternative sets simply hit function key f9 (which will re-run the KEYSET program) and then hit the appropriate selection key (ie. TAB etc).

To actually run your program you could either execute, in immediate mode, the command GOTO<start line of your program>, or you could delete the KEYSET routine from the beginning. The first of these two options is preferable in that you don't end up losing the routine and hence have to perform a 'merge' to get it back.

By changing the KEYSET program you could permit more sets to be available or change the assignment of the keys, or both. A well thought out range of assignments can save an awful amount of time during the typing in of a program and greatly reduces the chances of a keyword typing error.

MULTIPLE KEY SELECTION WITHIN A PROGRAM

This method is easy to perform on a series one (eg: OS 1.2) operating system, but is a little complex on OS 0.1. We will give the OS 1.2 solution first, followed by brief notes on how to implement a version for earlier systems.

The new operating system has a set of FX calls (*FX225-*FX228) to enable the so-called 'base' number of the function keys to be set. When you

press the function keys together with either SHIFT, or CTRL, or both, a character code is generated. These codes are normally set to give useful Teletext control codes (see hints in BEEBUG vol.1 no.9 p.31 or, in Mode7, try SHIFT f3 followed by some text - it will appear in yellow). The FX calls mentioned above can be used to determine which character codes the keys produce.

The skeleton program below uses these calls to set the keys so that they will give characters with codes from 140 to 179. This is achieved in PROCInitialise. PROCFunctionKeySelect sets up a loop with the GET function and searches for one of these codes - given by a function key on its own, SHIFT function key, CTRL function key, or SHIFT/CTRL function key. A corresponding value for variable 'K%' between 1 and 40 is then generated and returned. The ON GOSUB selection statement at line 200 in the main program then directs execution to one of 40 subroutines placed at lines 3100 onwards:

```
100REM This program shows you how to
110REM select up to 40 routines by
120REM using the function keys.
140REM By T F Powys-Lybbe & BEEBUG
150:
160REM MAIN PROGRAM
170PROCInitialise
180REPEAT
190PROCFunctionKeySelect
200ON K% GOSUB 3100,3200,3345,.....,4000
210UNTIL finished=TRUE
220END
230:
1000DEF PROCInitialise
1005finished=FALSE
1010*FX225,140
1020*FX226,150
1030*FX227,160
1040*FX228,170
1120VDU15:CLS
1140ENDPROC
1150:
2000DEF PROCFunctionKeySelect
2020REPEAT:K%=GET:UNTILK%>139 ANDK%<180
2025K%=K%-139
2050ENDPROC
2060:
3000REM INITIAL STAGE SUBROUTINES
3100REM f0 command
3190RETURN
3200REM f1 command
3335RETURN
3345REM f2 command
3500RETURN
3510:
3520:
4000REM f9 and <shift> command
4010 PRINT"Are you sure you want to exit?
(Y/N) ";
4020REPEAT:A$=GET$:UNTILINSTR("YNyn",A$)>0
4050IFINSTR("Yy",A$)>0 finished=TRUE
4060PRINTA$'
4300RETURN
4310:
5000REM PROCEDURES & FUNCTIONS
5010:
5020:
```

If you have the earlier 0.1 operating system then you can achieve a similar effect by first defining each of the 10 keys, in PROCInitialise, to produce a single character as follows:

```
*KEY0 |!|L
*KEY1 |!|M
*KEY2 ...etc
```

These are detected by the same form of REPEAT loop in PROCFunctionKeySelect as used above, with the GET statement, but using limits of 139 and 150 instead. The simultaneous pressing of SHIFT, CTRL etc is then detected by a set of negative INKEY functions. The variable K% is assigned a value between 1 and 40 by subtracting 139 (as above) and adding on either 10, 20, or 30 depending on the results of the INKEY tests. From this point on the program would proceed as for the OS 1.2 version.

Although only a skeleton program is shown it is easy to see how the concept can be used in practice. Using function keys instead of having to type in options (particularly 'word' options) during the course of using a program is very neat and quick - try it and see.

CENTRING JOYSTICKS

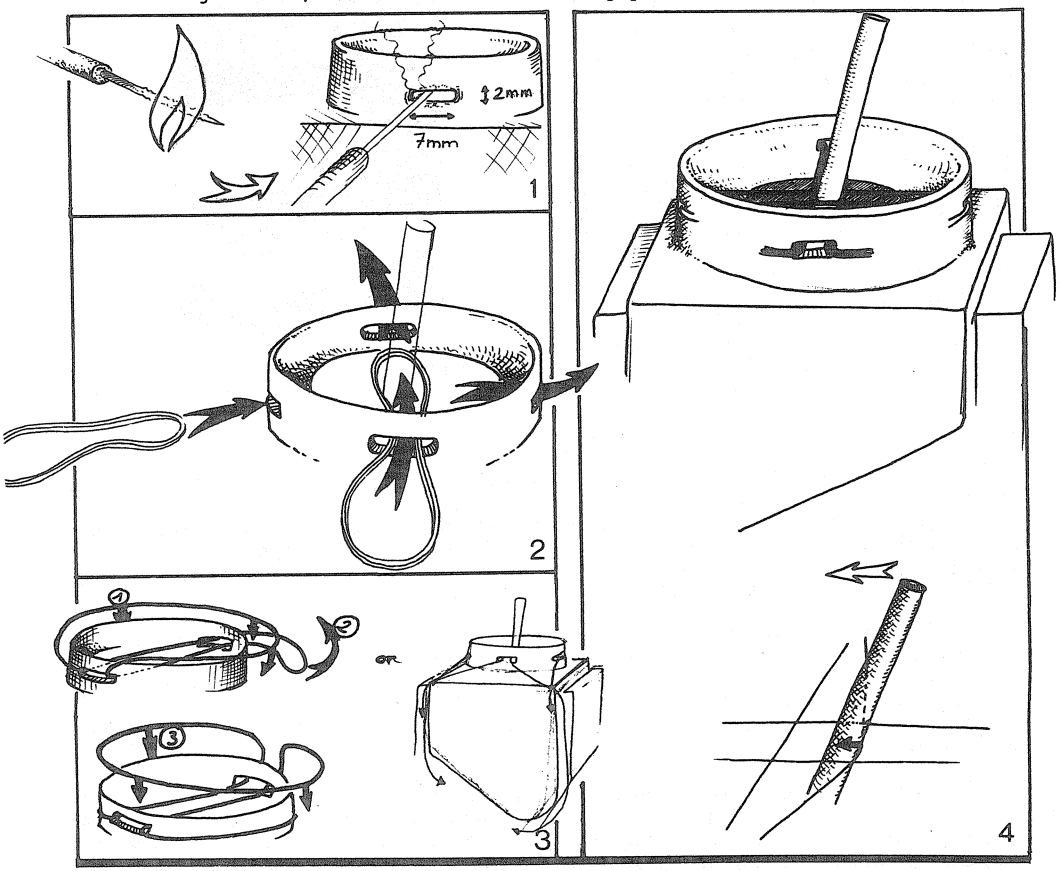
by F. Laude

Frederic Laude, a French member, provides a series of sketches explaining how he obtains auto-centring ANHØ1 joysticks.

For certain programs that use joysticks the ideal type of joystick would be one which auto-centres. The ANHØ1 model is widely used amongst Beeb owners but it does not auto-centre, that is until now!

The drawings themselves are so clear that they need very little further comment. Here however are four points worth noting:

1) Make sure that the implement you use to make the slots has a well insulating handle, or it will be more than the plastic which gets burnt.
2) Try not to inhale too much of the fumes given off - they can be pretty obnoxious and may even be toxic.
3) Make sure the slots are big enough for good strong elastic bands. The two options shown for 'locking' the bands in place will depend more-or-less solely on the length of the bands.
4) When making the slots be careful not to stick the heated 'poker' too far in as you may end up with a drooping joystick lever!



BBC BOOK REVIEWS

Book: The BBC Micro - An Expert Guide (155 pages)
By: Mike James
Pub: Granada Publishing Ltd
Price: £6.95 paperback
Reviewer: Colin Opie

The book is certainly not aimed at the newcomer, and it covers a lot of ground, as can be seen by a list of the contents:

BBC Computer Hardware	BBC Basic
Machine Operating System	Video Display
Sound Generator	Interfacing
Intro to Assembly Language	Assembly Language II

Mike James does point out in the Preface that: "..even in a book of this length there must be more unsaid than said", and my initial question was whether or not a book of 155 pages, (which I do not consider to be as long as Mike James implies), can actually be sub-titled 'An Expert Guide'?

In general I did find that the individual chapters supplemented the User Guide very well. Both hardware and software are discussed, for each of the topics covered, where this is applicable. Coverage of VIA, ACIA and ADC port addresses and programming was thorough, although it was largely assumed that 'trying things out' was the responsibility of the reader's imagination. An example of how he does supplement the User Guide is found in the chapter on Basic. This is in two main parts: a) How is it different from other Basics? (eg. Procedures, Repeat-Until), and b) Explanations on how the Interpreter runs, how programs are stored, Garbage Collection etc.

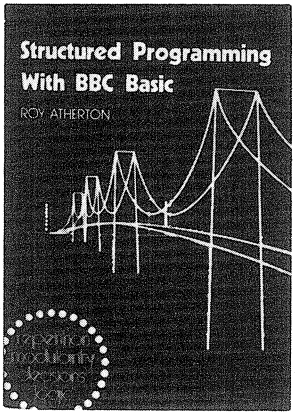
For the most part Mike James' explanations are thorough, and I found the chapter on the Sound Generator to be particularly clear in this respect. The chapter on Interfacing is restricted to the Analogue inputs and the User Port, but within these confines the text is again quite clear and readable. The two chapters on Assembly Language give a good grounding for starting work in this area, and there are a few 'worked' projects as examples of Assembly programming.

From a production point of view I thought it a shame that the book had no appendices for quick reference. This is particularly pertinent in that the Contents and Index are a bit skimpy. Program listings are clear and well presented but still printed by a dot matrix printer. I realise this is one of my hobby-horses but it is a pity as the rest of the book is very clearly set out, with a good quality print-face.

In conclusion, and indeed to answer my own question, I would say that the book is too small to be a true 'expert guide' (such a guide would more likely be a suite of books). However it is not a 'novice guide' but rather an excellent text for the

person who is getting to grips with BBC computer terminology and who wants to expand his knowledge without having to break the bank. The Basic programmer can buy it and learn a bit more about hardware (or Assembler), and vice versa. It is good value for money and should be a welcome addition to the owner's library.

Book: Structured Programming With BBC Basic (207 pages)
By: Roy Atherton
Pub: Ellis Horwood/Heinemann
Price: £6.50 paperback or £12.50 hardback
Reviewer: Sheridan Williams



Roy Atherton is perhaps the arch proponent of structured programming languages, and has long been a critic of Basic. BBC Basic is, however, in a different class from 'ordinary' Basics, being endowed with many of the features found in languages like Comal and Pascal.

The theme behind this book is to encourage the processes that PRECEDE programming - problem analysis and design. With this aim in mind it is not a book to be taken light heartedly, rather, it is a very serious book with a lot of very good material in it. There are many worked examples and solved problems. The book uses BBC Basic in the majority of its examples, making it very easy to work through.
I thoroughly recommend this book to those of you who want to take the art of programming seriously; it will teach you the CORRECT way to program.

The book itself, is cleverly structured making it of use to both the beginner (before he picks up bad programming habits) and the experienced programmer who is ALREADY practising bad habits. A glance at some of the chapter headings will show you that this is no ordinary book on programming:

The Computing Process
Data Representation and Manipulation
Shapes and Procedures
Decisions and Special Characters
Logic, Colour and Control
Procedures, Parameters and Functions
Data Structures
The Anatomy of a Program

In conclusion the author gets my greatest admiration for a book that is totally devoid of those dreadful GOTOs. If everyone wrote programs like this the job of understanding and debugging would be vastly easier.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SAVING SPACE IN FUNCTION KEY DEFINITIONS

Malcolm Inglis offers the following tip for publication in your magazine. You can save space in the function key buffer by using BASIC tokens rather than source. 'Broken bar/exclamation mark' in *KEY definitions causes &80 to be added to the ASCII value of the immediately following character, so any required token can be generated by |! followed by the appropriate character. For example:

*KEY0 |!q|!3|M

occupies only 3 bytes and may be used instead of *KEY0 PRINT RND |M

ROTATING AND EXPANDING CHARACTERS

by P Manhire

Program tested on O.S. 0·1 and 1·2

This is a collection of short programs in assembly code which will manipulate any printable character (alphanumeric or user defined) in a variety of ways. These procedures can be used in many ways, for example:

(1) Produce quadruple sized text in any mode (except mode7)
(2) Producing new character definitions (for games etc.). It is possible to produce giant invaders or inverted frogs or rotating munchy-men or whatever, from previous character definitions without having to work out the new definitions.
(3) Labelling the vertical axes of graphs, diagrams,etc.

The program listed at the end of the article contains two parts, the assembly code for the character manipulation - this is contained within the procedure PROCASS listed from 250 to 990. The procedure is called in line 70, and assembles the machine code into the RS423 buffer at &A00. You can change the address by altering line 350. The machine code program is subdivided into five routines which may each be called separately. The five calls and their functions are given in the table.

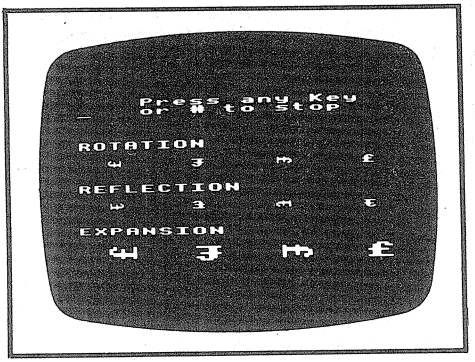
The remainder of the program, lines 10 to 240, sets up a demonstration of the routines, and when run, it results in any letter entered from the keyboard being printed to the screen in a variety of ways - reflected, rotated, inverted and expanded. Inspection of this program will show how to call and use the various routines, but here are a few additional examples:

1

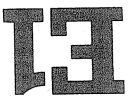
All operations must begin by setting X% equal to the ASCII code of the character to be manipulated (it can be one that you have defined yourself) and then calling Getchr. Use the BEEBUG reference card, page 5 to get ASCII codes - eg. ASCII 75 = "K". If you want to manipulate your own characters, first define them with VDU23 (see User Guide p.384 or BEEBUG reference card p.5), and then set X% to the character number that you have chosen (avoid values 250 to 254 since these are used by this program).

To take a real example, we will manipulate a space invader. Line 10 below creates this as character 224 (try executing VDU224 after this in modes 4 or 5). Line 20 stores the character in the locations reserved by the manipulator routine. To rotate the invader you simply call the routine 'Rot' (in line 30 below). The rotation and inversion routines 'Rot' and 'Invert' put the new character into character 250, so to print it, line 40 is used below:

```
10 VDU23,224,60,126,219,255,126,60,66,36
20 X%=224:CALL Getchr
30 CALL Rot
40 PRINT CHR$250
```



To turn it on its head, add:



```
50 X%=250:CALL Getchr
60 CALL Rot
70 PRINT CHR$250
```

This picks up the rotated invader, and rotates it once again.

The following sequence will produce an expanded version:

```
X%=224:CALL Getchr:CALL Expand
```

The easiest way to print it is to define a variable say 'character$' as follows:

```
character$=CHR$251+CHR$252+CHR$10+CHR$8
+CHR$8+CHR$253+CHR$254
```

then PRINT character$ will print the expanded character, and this can be TABbed to anywhere on the screen.

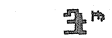
Here are two further examples:

```
X%=ASC"W":CALL Getchr:CALL Rot:PRINT CH
R$250
```

would print a letter "W" on its side.

```
PRINT"`":X%=ASC("`"):CALL Getchr:X%=250
:CALL Rot:CALL Getchr:CALL Expand: PRIN
T CHR$251;CHR$252'CHR$253;CHR$254
```

would produce:



NOTE 1: The code is loaded between &A00 and &AFF. This APPEARS to be safe and it leaves &D00 to &DFF free for a printer screen dump. If you do not trust this or you are using the RS432 interface you must relocate the code elsewhere. This is done by changing line 350 in PROCASS from P%=&A00 to P%=(new location)

NOTE 2: The code can be saved without the BASIC assembler. This is done by running the program then typing:

*SAVE"CODE" A00 B00 A00 <return>

The code will be saved on to tape or disc. To reload it directly into the memory type: *LOAD"CODE"

and press 'play' if you are using cassette. If you do this you will not be able to call the machine code procedures by name (Getchr etc). You must use the addresses in the REM statements in PROCASS instead. For example, use CALL &A5D instead of CALL Expand.

ASSEMBLER PROCEDURES

Getchr	This reads the dot pattern of any character into the memory at locations &71 to &78 and &81 to &88. To use it; first set X% to the ASCII code of the character you want to manipulate.
Putchr	This performs a VDU 23 call to define a character with the manipulated character in &81 to &88. To use it; first set X% to the ASCII code of the character you want to define.
Rot	This rotates the character read by Getchr ANTICLOCKWISE by 90 degrees. CHR$250 is always defined as the rotated character but Rot followed by Put will define any other character.
Invert	This inverts the character read by Getchr (ie. like the reflection in a pool of water). CHR$250 is always defined as the inverted character but Invert followed by Put will define any other character.
Expand	This takes the character read by Getchr and expands it to fill a block of four characters (always CHR$250 to CHR$254).

```
  10 REM"Character Manipulation
  20 REM"P.Manhire March 1983
  30 REM"expands,reflects and/or
  40 REM"rotates characters
  50
  60 REMDEMONSTRATION PROGRAM
  70 PROCASS
  80 A$=CHR$8+CHR$8+CHR$10:REM"BACK 2
 spaces,DOWN 1 space
  90 B$=CHR$251+CHR$252+A$+CHR$253+CHR
$254:REM"EXPANDED character
 100 MODE5:VDU19,2,5,0,0,0,19,3,6,0,0,
0
 110 PRINT TAB(4,5);"Press any Key":C$
=GET$
 120 IF C$="#" THEN MODE7:END
 130 PRINT TAB(0,10);"ROTATION"
 140 PRINT TAB(0,15);"REFLECTION"
 150 PRINT TAB(0,20);"EXPANSION"
 160 COLOUR2:X%=ASC(C$):CALL Getchr
 170 X%=250:FOR I%=0 TO 3:CALL Rot:PRI
NT TAB(2+I%*5,12);CHR$250:CALL Getchr:N
EXT
 180 X%=250:CALL Getchr:CALL Invert:CA
LL Getchr:FOR I%=0 TO 3
 190 CALL Rot::PRINT TAB(2+I%*5,17);CH
R$250:CALL Getchr:NEXT
 200 X%=ASC(C$):CALL Getchr:X%=250
 210 FOR I%=0 TO 3:CALL Rot:CALL Getch
r:CALL Expand:PRINT TAB(2+I%*5,22);B$:C
ALL Getchr:NEXT
 220 COLOUR3:PRINT TAB(4,6);"or # to s
top":GOTO 110
 230 REM"END OF DEM. PROGRAM
 240
 250 DEFPROCASS
 260 REM"Assembles code at &A00
 270 REM"Relocate if RS423 is used
 280 REM"CALLS NAME     ADDRESS
 290 REM"      Getchr   &A00
 300 REM"      Rot      &A17
 310 REM"      Putchr   &A33
 320 REM"      Invert   &A4D
 330 REM"      Expand   &A5D
 340 FOR J%=0 TO 3 STEP 3
 350 P%=&A00
 360 [OPT 0
 370 .Getchr
 380 \"****
 390 STX &70:STX &80:LDX#&70:LDY #0:LD
A #10
 400 JSR &FFF1:LDX#&80:LDY #0:LDA #10
 410 JSR &FFF1:RTS
 420 .Rot
 430 \"****
 440 LDX #&71:LDY #&88
 450 .Loop:ROL 0,X:TXA:STY &79:LDX &79
 460 ROL 0,X:TAX:INX:CPX #&79:BNE Loop
 470 LDX #&71:DEY:CPY #&80:BNE Loop
 480 LDX #250
 490 .Putchr
 500 \"****
 510 LDA #23:JSR &FFEE
 520 TXA:JSR &FFEE
 530 LDX #&81
 540  .Loop2:LDA 0,X:JSR &FFEE
 550 INX:CPX #&89:BNE Loop2:RTS
 560 .Invert
 570 LDX #&71:LDY #&88
 580 \"****
 590 .Loop3
 600 LDA 0,X:STA 0,Y
 610 INX:DEY:CPY #&80:BNE Loop3
 620 LDX #250
 630 JMP Putchr
 640 .Expand
 650 \"****
 660 LDX #251:STX&8E
 670 LDX #0:STX &8C:STX &8D
 680 LDX #&71:STX &8A
 690 LDX #&81:STX &8B
 700 JSR Qchr
 710 LDX #252:STX&8E
 720 LDX #0:STX &8C:STX &8D
 730 LDX #&71:STX &8A
 740 LDX #&81:STX &8B:JSR Qchr
 750 LDX #253:STX&8E
 760 LDX #0:STX &8C:STX &8D
 770 LDX #&75:STX &8A
 780 LDX #&81:STX &8B :JSR Qchr
 790 LDX #254:STX&8E
 800 LDX #0:STX &8C:STX &8D
 810 LDX #&75:STX &8A
 820 LDX #&81:STX &8B :JSR Qchr
 830 RTS
 840 .Qchr
 850 .Loop5
 860 LDX &8A:ROL 0,X:LDX &8B:JSR Enter
 870 LDX &8C:INX:STX &8C:CPX #4
 880 BNE Loop5:LDX #0:STX &8C:LDX &8A
 890 INX:STX &8A:LDX &8B:INX:INX
 900 STX &8B:LDX &8D:INX:STX &8D
 910 CPX #4:BNE Loop5:LDX &8E:JSR Putc
hr
 920 RTS
 930 .Enter:BCS Full
 940 CLC:ROL 0,X:CLC:ROL 0,X:INX
 950 CLC:ROL 0,X:CLC:ROL 0,X:RTS
 960 .Full
 970 SEC:ROL 0,X:SEC:ROL 0,X:INX
 980 SEC:ROL 0,X:SEC:ROL 0,X:RTS
 990 ]:NEXT:ENDPROC
```

Program tested on O.S. 0·1 and 1.2

USING FILES (Part 3)

by Sheridan Williams

Although many people manage for some time using cassettes, particular types of files and file processing requires discs. This part of the series of articles on files discusses a very important and major topic which is only really applicable to disc based systems.

The advantages of discs are twofold:

1. The program need not be constructed as one LONG program, but can be written as several shorter ones linked (CHAINed) via a main menu.
2. The whole file need not be read into main store in its entirety, but simply a single record at a time. This means that the only limiting factor on file size is the disc's capacity, and with the PEOPLE file defined last month, you should be able to hold about 4000 records even on a 100k disc.

However the way in which you process the file will be critical. If it is processed 'serially' then you will have to reduce this by at least a factor of two, unless you have twin disc drives or double sided drives. This is because a serial file requires that two files be open simultaneously, one for reading, one for writing; they are invariably of similar size.

I will now give details of how to append (add to the end) of a serial file. You should note that a serial file can only be opened for input or output, and not both at the same time. This means that you cannot simply read the file to the end and then write more data to it. The process is to create a new file and open it for output. Read a record at a time from the input file and then write it to the new file, when all the copying has been done you can then add any new records to the end of the new file before closing it. Once that is complete you can proceed to re-name the files so that your program can work next time it is run.

The following program will do this:

```
10 c1=OPENIN"PEOPLE":IF c1=0 THEN
   c1=OPENOUT"PEOPLE":CLOSE#c1:
   GOTO 10
20 c2=OPENOUT"N.PEOPLE"
30 IF EOF#c1 THEN 90
40  REPEAT
50   INPUT#c1,sur$,title$,sex$,dob$
60   PRINT#c2,sur$,title$,sex$,dob$
70   PRINT sur$" "title$" "sex$" "
     dob$
80  UNTIL EOF#c1
90 CLOSE#c1
100 INPUT"Surname",sur$: IF sur$=""
    THEN160
110 INPUT"Title  ",title$
120 INPUT"Sex    ",sex$
130 INPUT"Date of birth",dob$
140 PRINT#c2,sur$,title$,sex$,dob$
150 GOTO 100
160 CLOSE#c2
170 *DELETE PEOPLE
180 *RENAME N.PEOPLE PEOPLE
```

The reason for including line 10 is because if the file "PEOPLE" does not exist (probably because this is the first time that you have run the program) then it cannot be opened for reading. It is therefore opened for writing and then closed before line 100 is executed again, and now it obviously DOES exist although it contains nothing.

If you wish to use variable file names you could input them at the start, but would have to access the "Command line interpreter" as described in the article "Passing Filenames to the MOS" in BEEBUG vol 1 no 10 p 31.

Processing a file serially involves reading successive records from the file one at a time until we require to read no more. Suppose that we had a file of 3000 customers on a disc and they were held in alphabetical order, further suppose that we wished to examine the record for the customer named "Zulu Enterprises". As this record must be

virtually at the end of the file we will have to read nearly 3000 records until we read the one that we want. You should note that whereas it wouldn't take long to read 3000 array items, it will take several minutes to read 3000 disc records, and this time might be prohibitive in many situations.

One answer is to use the PTR# statement which allows you to control where in the file to read/write next. For example PTR#c=1234 will instruct the file pointer to move to the 1234th character in the file. Beware though, because you dare not execute an INPUT# from this position because you may have moved the pointer into the MIDDLE of a number or string on the file, and INPUT# will then give an error. There is another problem, how do you know where to move the pointer to gain access to the 2994th record say? All this sounds very vague so let us immediately move on to two other main methods for processing files.

DIRECT ACCESS FILES

Type 1 - Random access

A random access file is one where you can move directly to any record and read, alter or write a record at that position. When the file is opened it is available for reading or writing or both. These types of files must really have fixed length records in order that you can calculate the start address for any record. If the records were not of fixed length then the act of overwriting one record with another that is longer would erase part of the succeeding record!

Type 2 - Index Sequential Files

This type of file is really a combination of two files - the 'index' file, and the 'data' file. When a record is required to be accessed the index is looked up first to find the address of the record on the data file. It is a combination of serial and direct access, and most frequently the index file is read into main store (into an array) in order to reduce the access time. If the index is too big to fit into main store in its entirety, then it either has to remain on disc, or just part of the index is read into main store. The index comprises only the 'key' field and the address of the record on the main file. This requires extra storage space on disc but this penalty is offset by the speed of access. If the index was in main store then any record could be accessed in less than 1 second. Another advantage is that records may be of variable length, and this may save more space in the file than the index itself occupies, resulting in an index sequential file occupying less space than its random access equivalent.

PRACTICAL USES OF RANDOM ACCESS FILES

Using the "People" file whose record description I gave in part 1 I will go through an application that performs the following processes on the file:

1. Initialise the file.
2. Read, write, amend and delete records on the file.
3. Search for records that match a particular specification.
4. Sort the file on disc.
5. Access a record by 'person' number, and surname.

To save referring back to part 1, here is the record description together with the suggested field lengths:

FIELD NO	FIELD NAME	VARIABLE NAME	LENGTH
1	Surname	sur$	15
2	Title	title$	10
3	Sex	sex$	1
4	Date of Birth	dob$	6

Referring to page 330 of the User Guide we find that integers take 5 bytes (5 characters) on the file, real numbers take 6 bytes, and strings require their length+2 bytes. There are two ways in which we may write data to the file:

(a). Join all the fields using rec$=sur$+title$+sex$+dob$, and then PRINT#c,rec$

(b). Write each on separately using PRINT#c,sur$,title$,sex$,dob$

Method (a) has to be preferred because a saving of 6 characters

results. This is because a string is stored taking up two characters more than its length, so in method (b) we write 4 strings whereas in method (a) we only write one. You may prefer method (b) as it is more straightforward, but just suppose you were to be storing a 2000 record file, you would waste 12k of file space; on a 200k disc this would cost you over 800 records!

With random access files we can still write records onto a file in a serial manner by successively PRINTing each record, however we CAN write a record anywhere on the file, thus we ought to give each record a number which reflects its position (record address) on the file. This is one reason why it appears that everything that is computerised seems to be given a number, for example payroll number, gas account number, credit card number, bank account number etc. The number often reflects the position of the record on the file.

Part four continues next month. There will be a demonstration program that manipulates a random access file, together with notes on random file updating and buffers.

WORDWISE UPDATE

by G. Foot

With reference to the article in BEEBUG vol.1 no.10 p.41 on 'Configuring Wordwise for Epson Printers' there are a number of hints which may be useful to be know about.

1) If through editing or some other means, spaces have been inserted at the end of the key definition lines in the program, then peculiar things can happen. For example a mysterious indent or interval in the formatted text can occur (depending on where the embedded command is employed when using Wordwise). To make sure that this doesn't happen check that no spaces exist at the end of the Basic program lines by using the hint given in BEEBUG vol.2 no.1 p.25.

2) When printing out text which uses the 'enlarged characters' mode of the Epson, problems may be encountered over getting text in the right place, especially when using the 'CE' command. At times the only way around this is to do a manual adjustment; but a good starter is to insert an embedded command which will halve the line length. The revised commands (assuming a line width of 62 characters) then become:

```
*KEY7 |!!OC27,87,1|!!LL31|!"
*KEY8 |!!OC25,87,0|!!LL62|!"
```

Notice that different commands have been used as these are more satisfactory than OC14 and OC20 (used in the original article), which are cancelled by the line feed.

3) For those using Wordwise with a TV set, include the command *TV0,1 in the Basic program to remove interlacing and hence obtain a steady picture in the 80 chars/line mode.

4) When using single sheets, eg. in letter writing, it is convenient to switch out the 'Paper Out' warning signal of the printer. This can be done by switching one of the internal DIP switches to the 'OFF' position, (Pin 6 of DIP switch 1 - see Epson manual), or by inserting the lines:

```
24 REM Disable out-of-paper alarm
26 VDU 2,1,27,1,56,3
```

into the Basic program.

FX CALL UPDATE

This list of new *FX calls, largely derived from Acorn, extends the listing given in BEEBUG vol 1, no 10, p 15, in an article on the new operating system.

*FX13 As given in User Guide but X=7 disables RS423 receive error event, and X=8 disables service/network error event.

*FX14 As given in User Guide but will also enable events given in *FX13 above.

*FX117 Returns VDU status byte in the X register.
BIT 0 - set if VDU2 sent, cleared by VDU3.
BIT 2 - set if 'paged mode' on, clear if 'paged mode' off.
BIT 3 - set if software scrolling (there is a text window), clear if hardware scrolling (no text window).
BIT 5 - set when cursors joined by VDU5.
BIT 7 - set if VDU disabled.

*FX118 Returns with carry bit set if CTRL key is pressed. Acorn claims it will return with negative bit set if SHIFT is pressed, but this doesn't seem to work.

*FX123 Used by the User Print Routine to indicate to MOS that it has finished its task. See *FX5,3 in User Guide.

*FX138 Now expanded to insert a character into any buffer, X contains buffer number (from *FX21 on p428 of User Guide), Y contains the character to be inserted. Useful codes to insert into the keyboard buffer are :-

139	COPY
140	Left arrow
141	Right arrow
142	Down arrow
143	Up arrow
128+fn key no.	(inserts function key).

*FX141 Equivalent to *ROM, select ROM cartridge system.

*FX142 Select language ROM, X contains ROM number, sockets are numbered 12,13,14,15 on main board. Using this on an empty socket or filing system ROM crashes the system.

*FX152 Examines buffer, buffer number in X. Returns with carry bit set if buffer empty, clear otherwise. According to Acorn Y contains the next character to be read (without removing it from the buffer) but Y does not seem to contain anything meaningful.

*FX153 Inserts a character into a buffer as for *FX138 but can only be used for X=0 or 1 and will generate an escape condition if the escape character (set by *FX220) is inserted.

*FX154 Write to video ULA (&FE20).

*FX155 Write to video ULA (&FE21).

*FX156 Change 6850 control register - register becomes (Old value AND Y) EOR X. See 6850 data sheet for more information.

*FX158 Read speech processor.

*FX159 Write speech processor. See Speech System User Guide for more information.

Important note - all FX calls greater than 166 write to &190+Call number with the standard form (Old value AND Y) EOR X so read them with X=0 and Y=255.

*FX179 Reset for OSHWM.

*FX180 Current OSHWM (Used by BASIC to initialize PAGE).

*FX189 Number of ADC channels (as set by *FX16).

*FX194 Flash period 2 (as set by *FX10 but this reads the value).

*FX195 Flash period 1 (as set by *FX9 but this reads the value).

*FX202 Set shift and caps lock.

both off	= &30
caps lock	= &20
shift lock	= &10
both on	= 0.

*FX211 Channel for CTRL-G. Default 1.

*FX212 Envelope for CTRL-G use (Envelope Number * 8) - 8. Default 144.

*FX213 Pitch for CTRL-G. Default 101.
*FX214 Duration for CTRL-G. Default 7.
*FX233 As *FX231 (User Guide p.441), but affects system 6522, use with extreme care!
*FX235 Return presence of Speech Processor.
X=&FF Speech processor present,
X=&00 Speech processor not present.
*FX241 Read/Write *FX1 value.
*FX242 Cassette motor status, off &64, on &E4.
*FX243 Offset to address of real time clock, contains 5 or 10, clock stored at &28D + offset.

*FX

Acknowledgement: We are grateful to Acorn Computers Ltd for permission to publish certain *FX calls listed above, and to Chris Bingley, Clive Harris, Matthew Rapier and William Lea for discovering and checking some extra ones.

A FEW EXAMPLES

As an example of the use of FX211 to 214 try this:

```
ENVELOPE3,&86,&FF,0,1,3,1,2,&7F,&FF,
&FD,&FD,&7E,&78
*FX211,0      Select Noise
*FX212,16     Envelope 3
*FX213,7      Pitch 7
*FX214,10     Duration 10
```

Then execute VDU7 or CTRL-G

To try FX138:

```
*KEY0,HELLO THERE
*FX138,0,128   Insert f0 into key buffer.
```

*FX

Also try -

```
*FX153,0,27    Insert escape character into key buffer.
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SOFTWARE DISPLAY SWITCH

To switch off the TV or monitor display (via software!), hence simulating a switched-off machine, M.A. Ridyard suggests that you execute the following:

```
VDU23;0;0;0;0;
```

This does not affect your program which can be recovered as per a BREAK start, or by changing Mode.

[This version of the VDU command effectively switches the display off by programming the video controller to use zero characters per line! Anything you type in will therefore be accepted, its just that you can't see it. Ed]

UNRESPONSIVE EPSON

If your EPSON MX80 type III does not seem to respond to the control codes such as ESC G (double character printing mode), try issuing the command as:

```
VDU 1,27,1,70
```

The VDU 1 command sends the next character to the printer only. This technique seems to solve the problem, though you must precede it with VDU 2 on the new operating system. Bob Tennent.

TEXT SEARCH

To test if a string occurs within another string M.J.Davies suggests that you use the Basic statement INSTR. The syntax is as follows:

eg. IF INSTR(input$,test$) THEN PRINT"Found"ELSE PRINT"Not Found"

where input$ is the string that you are searching and text$ is the string that you wish to find.

[This is the normal use for INSTR, but we felt it worth pointing out. Also note that if you have BASIC I there is a bug in INSTR in that the first string must be longer than the second string, or the system will crash. Ed]

NEW OPERATING SYSTEM ROUNDUP

A number of points have come to light either through experience or as a direct result of upgrading to the OS 1.2, as described below by various members.

CONFIGURING THE RS423

a) Alan Bray has a method of reconfigering the RS423 to accommodate different printers. The method involves altering location &FE08, which is the control byte for the RS423, by using the *FX call: *FX 151,8,value where:

value =2 gives 7 data bits:even parity:2 stop bits.
=6 gives 7 data bits:odd parity :1 stop bit.
=10 gives 7 data bits:even parity:1 stop bit.
=14 gives 7 data bits:no parity :1 stop bit.
=18 gives 8 data bits:no parity :2 stop bits.
=22 gives 8 data bits:no parity :1 stop bit, (Default setting).
=26 gives 8 data bits:even parity:1 stop bit.
=30 gives 8 data bits:odd parity :1 stop bit.

b) J.C.Price has just had OS 1.2 fitted and discovered that the Epson MX 80 F/T no longer functioned correctly via the serial RS423 port. The solution is *FX 156,16,227, and a letter from Acorn is given below detailing this:

"...In your case it is necessary to set 2 stop bits on the RS423, to do this you should use: OSBYTE with A=&9C, Y=&E3 and X=4*(value from figure 1), ie. for 8 bits and 2 stop bits: X=&10 or *FX156,16,227. Y then sets a mask and the word select bits CR2, CR3, CR4 are changed with an appropriate value in X taken from the data sheet."

[Ed. Both methods described above alter the 6850 Control register dealing with the RS423 port. The second method implies access to the appropriate data sheet (see *FX call update elsewhere in this issue), and is a little more complicated to use than the first method discussed.]

VECTOR AT &D00 WARNING

Cedric Marshal warns us that the OS 1.2 now vectors the NMI on to &0D00. This is potentially very useful, but BEWARE - using BREAK will over-write the first byte of any code at &0D00 with the instruction RTI!

VDU 21 PROBLEM

An anonymous person in Northumberland has just completed the task of modifying OS 0.1 programs to run on OS 1.2. He writes that this should not have been necessary in an upwards compatible operating system but the major problem encountered was that VDU 21 no longer operates correctly - it does disable screen drivers but any CR code executed is sent straight to the printer even if the printer is disabled with CTRL C (if your printer has auto line feeds try pressing return key after executing VDU 21).

[One way around the problem is to replace VDU 21 with VDU 28,0,0,0,0. This may be cancelled with VDU 26. Ed.]

*FX3 BUG?

The *FX3,N call can be used with different values of N to route output to the screen, printer or RS423 in various combinations. These are detailed on page 422 of the User Guide. However, *FX3,3 does not work. This is a potentially useful call in that it is supposed to route output to the printer, but not the screen. Acorn tell us that the fault is in the User Guide rather than the new operating system !

(When is a bug not a bug ?). In fact their quite plausible explanation is that *FX3,3 disables the VDU drivers, and since the print routine works through the drivers, it unfortunately disables the printer as well.

There are various ways around the problem. You can in fact use *FX3,10 (using the undocumented bit 4 - see U.G. p 422). This turns off the VDU drivers, and therefore stops data going to the screen, but it sends data to the printer, bypassing the VDU drivers. One problem with this is that it does not filter out control codes, so that odd effects can occur on your printer if you print control codes. For this reason you do not need to execute VDU 1 (or VDU 2) to get control codes to the printer. *FX3,10 is cancelled by *FX3. An alternative approach is to execute VDU 2,21 which first enables the printer, and then disables screen printing without knocking out the printer. To return things to normal, use VDU 3,6 which turns off the printer, and re-enables screen printing. DEG

DISC PROGRAM RELOCATOR (32k)

by Allan Smith

Having seen the hint in BEEBUG (Feb 83) on moving large cassette programs down the memory, so as to accomodate them on disc machines, Allan Smith has devised a way of performing the whole operation automatically. For those who did not see the earlier hint, the problem is that many cassette-based programs are too long to run on a disc machine. This technique gets around the problem (unless the original program was designed to use ONLY mode 7).

MOVING DOWN & AUTO-RUN

Type *TAPE (but do not change PAGE), and load your program from cassette. DO NOT try to run it, but simply add the following lines to it (this itself can be mechanised - see BEEBUG vol 1 no 9 p 20 on merging programs):

```
 1 *KEY0 *TAPE|MFOR I%=0 TO TOP-PAGE STEP 4:
  I%!&E00=I%!&1900:NEXT|MPAGE=&E00|MEND|MLOMEM=TOP|MGOTO10|M
 2 *FX 138,0,128
 3 ON ERROR DELETE 1,4
 4 ERROR
10 REM Rest of program
20 :
```

The whole program should then be saved to disc (don't forget to type *DISC). Now, when CHAINed in the program will automatically load, relocate itself and run.

Notes:
a) line 1 is the same as the original tip except 'f0' and GOTO 10.
b) line 2 activates function key 0 automatically by using the new *FX call (see p.433 of User Guide).
c) lines 3,4 delete lines 1 to 4 once they are of no further use.

If you have a program residing at PAGE=&E00 that you wish to move up to &1900 for saving on disc, the following may be used. The routine is essentially the reverse of that described above.

```
*KEY1 FORB%=TOP TO PAGE STEP-1:?(B%+2816)=?B%:
NEXT B%|MPAGE=&1900|MEND|MLOMEM=TOP|M*DISC|M
```

Then press function key f1. It is probably worth using *SAVE to keep an 'invisible' copy of this key routine. See BEEBUG vol.1 no.5 p.26 for details.

SEIKOSHA GP100 SCREEN DUMP REVISITED

by Terry Bromilow

In BEEBUG vol.1 no.9 p.8 there was an article on machine code screen dumps for the EPSON (MX) and SEIKOSHA (GP80/100) printers. Some members have written asking for the assembly listing of the SEIKOSHA version to help with debugging their copying errors and to learn a little more about machine code programming. To be fair to Terry Bromilow he did in fact provide a fully documented assembler version, (the magazine copy was edited and only showed a series of DATA statements holding the machine code values).

To incorporate the code into the original copy perform the following:

a) LOAD in the original program as given by the listing in the February issue of BEEBUG (p.9).
b) DELETE lines 29130 - 29420
c) AUTO 29130,5 and enter the lines as given below. (Performing an AUTO will save you having to type in the five-figure line numbers).
d) SAVE this new copy of the program on a different tape or as a different file on disc.

If you change the argument of OPT in line 29205 to A% (as suggested by the corresponding REM statement), and take out the call to the dump procedure in line 29080, you can obtain a listing of the assembled machine code without performing an actual 'dump' by doing a dummy RUN.

This month's magazine cassette will contain a complete copy of the program.

```
29130 P%=TOP-174
29135 Y%=P%:        REM Delete this lin
e and all up to **** after first use
29140 P%!-2=&1503:  REM Disables printe
r and VDU if the m/code line is LISTed
29145 OSWRCH=&FFEE: REM Call address eq
uiv. to VDU statement
29150 OSWORD=&FFF1: REM Call address us
ed in POINT statement
29155 LINADD=&84:   REM Start of line b
eing printed
29160 GBYTE =&86:   REM Byte being cons
tructed to send to printer
29165 COLOR =&87:   REM To select requi
red colours
29170 XADD  =&88:   REM X address on sc
reen for POINT call
29175 YADD  =&8A:   REM Y address on sc
reen for POINT call
29180 OUTPUT=&8C:   REM Output of POINT
 routine call
29185 BITCT =&8D:   REM Bit counter
29190 NULCT =&8E:   REM Blanks counter
29195 A%=0
29200 REPEAT
29205   P%=Y%: [ OPT A%                \ m
ake OPT A% for listing
29210 STY LINADD: STA LINADD+1      \ Y
address of line
29215 LDX #0: STX XADD: STX XADD+1 \ X
address for POINT call
29220 STX NULCT: STX NULCT+1
29225 INX: BNE nexbyte              \X r
eg always=1 (except in POINT call)
29230 .send TXA: JSR OSWRCH: LDA GBYTE:
 JSR OSWRCH        \ equiv to VDU1,GBYTE
29235 .step LDA XADD: CLC: ADC #4: STA
XADD: BNE nexbyte \ step to next pixel
29240  INC XADD+1: LDA XADD+1: CMP #5:
BNE nexbyte       \ &500=1280
29245 TXA: JSR OSWRCH: LDA #13: JSR OSW
RCH: RTS          \ send endofline
29250 .nexbyte LDA LINADD: STA YADD: LD
A LINADD+1: STA YADD+1
29255 STX GBYTE                     \ set
GBYTE bit0 (this will roll up to bit 7)
29260 LDA #7: STA BITCT             \ Coun
t seven pixels (28 POINTs)
29265 .byte LDA #9: LDX #XADD: LDY #0:
JSR OSWORD     \ POINT subroutine
29270 LDA OUTPUT: CLC: BMI bit      \
collect result. Branch if off screen.
29275 AND COLOR: BEQ bit:NOP:NOP:SEC \
branch if not reqd colour
29280 .bit ROL GBYTE                \
roll into byte
29285 LDA YADD: ADC #4: STA YADD:BCC co
unt:INC YADD+1 \ step Y address
29290 .count DEC BITCT: BNE byte
              \ six more bits
29295 LDX #0: INX: LDY #&80
              \ for convenience
29300 CPY GBYTE: BEQ nul
              \ if all 7 bits zero
```

```
29305 LDA NULCT: .sender BEQ send
                \ if no stored nuls
29310 BNE sendnuls: NOP
                \ else send them
29315 .nul INC NULCT: BNE step
     \ inc nul-count - branch if <256
29320 INC NULCT+1: BNE step
     \ else step hi-count
29325 .sendnuls TXA: JSR OSWRCH: LDA #2
8: JSR OSWRCH
29330
     \ equiv VDU1,28 (rpt-grp follows)
29335 TXA: JSR OSWRCH: LDA NULCT:JSR OS
WRCH \ VDU1,nul-count
29340 TXA: JSR OSWRCH: TYA: JSR OSWRCH
     \ VDU1,&80(nul)
29345 LDA #0: STA NULCT
     \ reset nul-count
29350 DEC NULCT+1: BPL sendnuls
     \ if NULCT+1 >0 send 256 more
29355 INC NULCT+1: BEQ sender
     \ zero NULCT+1, indirect branch
29360 ]
29365 A%=A%+3: UNTIL A%>3
29370      ?P%=6: REM To re-enable VDU w
hen LISTing
29375      P%=Y%: REM Last line to be de
leted after first use        ****
```

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

TV RADIATION

Some time ago, in another journal, Sheridan Williams asked his readers if there was any risk from the radiation given off by a colour TV tube. I am an anaesthetist, not a radiologist, but I think that I can reassure him.

There is in fact a British Standard (BS 415) which limits the radiation to less than 0.5 millirem (sic!) per hour, at 5 centimetres from the screen. All domestic TVs tested in 1970 emitted radiation less than 1/50th of this standard, and the inverse square law will apply to viewing distances. A typical value might be 0.2 microrem/hour at 2.5 metres from the screen, which should be compared with the several hundred microrem received everyday from natural, background radiation.

Claims of other effects, due to "optical radiation" or to static electricity, are unconfirmed. In my own view, symptoms of malaise, headache etc associated with VDUs are due to incorrect seating and poor lighting, or to badly adjusted TV/monitor screens.

References:

1. British Medical Journal
   1981, Vol. 283, p. 1590
2. B.M.J. 1982 Vol. 285
   p. 282

[I have received dozens of replies on this topic all giving the same sort of reply. Thanks to all the people who wrote. I feel the matter can now be safely closed! SW]

EIGHT MILLION AT PLANETOID?

Astonished to say the least at John Benfield's eight million an Acornsoft Planetoid, I disassembled this masterpiece of programming to discover that the score is held as a BCD number in 3 bytes; ie its maximum value is 999,999. Admittedly I have had my copy of Planetoid (Defender) for a rather long time, and for new releases there may be scope for higher scores, but if Mr. Benfield has had his copy long enough to score 8,000,000 on it, it must be just as old as mine. Mr. Benfield is obviously blissfully unaware of this fact, perhaps it would be a good idea for you to comfort other Planetoid players who had given up all hope of ever becoming this good at the game. Phylyp Abercrombie.
[We offer John Benfield the right of reply. Ed.]

PRINTER QUERY

Zoe Hoyle is trying to use the *FX5,3 facility with a user supplied printer driver but finds that the computer locks up after about 50 characters even when the device is a simple RTS (Ready-To-Send). Does anyone know the correct way to use this facility ?
[There is not really enough information here to be able to comment, but the fact that the Beeb locks up is due to it not getting a valid 'NOT BUSY' signal FROM the printing device. Unfortunately the Beeb serial interface does not default to a 'printer not busy' state, and therefore a BUSY line must be provided and permanently wired 'inactive' for a simple 'Ready To Send' device. Ed.]

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

GOOD MAGAZINE

I've received the OS 1.2 and the Wordwise chips: no problems in installing them: am highly delighted with Wordwise. Thank you for recommending and supplying!

The magazines continue to be interesting and I find I refer to them as constantly as the User Guide. But please go back to the old style of cover with clearer contents list: type too small and the blue overprinting too dense on the March issue.

Dennis Kemp.

[Thank you for the words of encouragement. We totally agree about the recent cover, and for this reason we have put the contents list on the first inside page of the magazine. Quite a number of members have written saying that they like the new cover, and we do not anticipate going back to the plain cover of the early issues. Ed]

IMPROVED RELIABILITY ON FERGUSON 3T07 RECORDERS

Luis Villazon has found a way of improving the reliability of the Ferguson 3T07 tape recorders. He writes:

"My tape recorder is the recommended Ferguson model 3T07 with the official modification to take a single cable with one 7 pin DIN plug on one end and a 5 pin plug on the other, and it was bought from an official dealer. Despite all this it took me quite a while to find an acceptable setting for the tone and volume controls and even then the reliability was very poor. My model B with disc interface uses a 1.0 MOS in EPROM and therefore it should be reliable but it certainly was not.

I noticed that the same recording could sometimes be LOADed correctly while other times it could not and after a lot of experimentation I have finally found a simple way of obtaining 100% reliability (at least so far, which means 2 months of daily use). My modification which I include in case other members suffer from the same disease is to solder a 3 ohm resistor between the ends of a 3.5 mm jack, and keep it permanently plugged in the earphone socket. In my case values above 3 ohms do not work, nor do values below 2 ohms (I tried 1, 2, 3 and 6 ohms). With this simple insertion, all tapes - commercial or home produced can be LOADed and SAVEd with the same settings, ie. TONE at max. and volume within a wide range, but I keep it very near the minimum".

If you are experiencing some difficulty with this recorder then perhaps the above information will be all you need to set things right."

SAVING TAPE PROGRAMS ONTO DISC

John D Towns writes: "I have a problem I would like to address to you. It relates to having a disc interface fitted and my desire to save onto discs various software, all of which is commercially produced.

I have a machine code fix which will move the programs when loaded down to the old default page &E00, so there are no immediate problems over memory size.

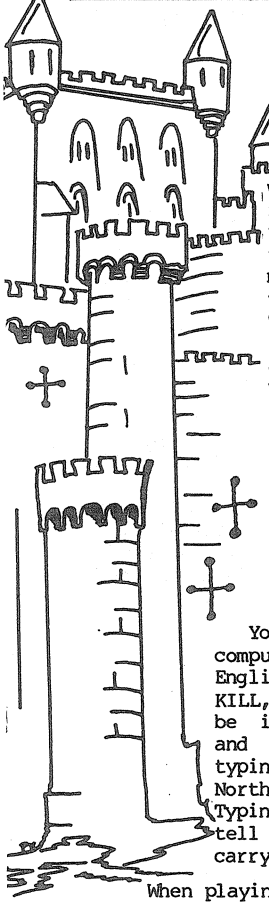
However the vast majority of the tapes have a machine code element. I have used the *OPT 1,2 command to locate the various addresses. Again the majority show a start address of FFF0E00 and execution address of FFFF801F. When I have examined these hex numbers as decimals, I find they have a minus sign in front. Why and what is the significance of this?

I have found it possible to *SAVE to tape using these locations, but cannot transfer to disc because the program will become corrupted. Also, a number of commercial programs are in two parts. I have not found it possible to *SAVE these either. I cannot break into them to check or alter the listings to run on disc, because I suspect they have CHAIN"" at the end of each part. Is there a way of solving my problems? Can you help?"

[Ed. See article on 'Secrets of Program Saving' by David Tall in v.2 no.1 p.18. This should help. Some programs are protected in various ways, and we cannot publish ways of defeating protected programs. One form of protection is simply to hide the start address of a machine code program, and you cannot successfully use a movedown routine without knowing the start address. The address &801F which you give is the entry address for Basic.]

RETURN OF THE DIAMOND (16k)

by R. McGregor



This is a nicely written adventure game for 16k machines. It is the basis of an article that will follow in next month's magazine about writing adventure games. The game itself is only a relatively small one as adventures go, but we will be telling you how to expand it next month.

It involves searching for a diamond which must then be taken to the "Diamond Castle". On your way you will meet some nasty enemies that are out to stop you.

You communicate with the computer through a series of English words such as GET, KILL, TAKE etc. Movement can be in any of four directions and this is accomplished by typing in N, S, E or W for North, South, East and West. Typing the word INVENTORY will tell you what you are carrying.

When playing this adventure you will probably need to construct a map to assist you in your travels. In case you get stuck, we will be publishing the map of this game next month.

```
10 MODE 7
20 PROCset up
30 PROCtitle
40 PROClook
50 REPEAT
60 REPEAT
70 INPUT"What now",c$
80 IF LEN(c$)=0 THEN PRINT"Eh?"
90 UNTIL LEN(c$)>0
100 PRINTSTRING$(39,"-")
110 PROCanalise
120 PROCtime_passing
130 UNTIL dead OR won
140 PROCfinish
150 RUN
160
170 DEF PROCset up
180 DIM place$(9)
190 FOR i=1 TO 9
200 READ place$(i)
210 NEXT i
220 DATA in a hut,in a garden
230 DATA in a shrubbery,on a path
240 DATA on a lane,in a forest
250 DATA at a dead end,at diamond cas
tle
260 DATA in a dark passage
270 DIM newpos(9,4)
280 FOR i=1 TO 9
290 FOR j=1 TO 4
300 READ newpos(i,j)
310 NEXT j
320 NEXT i
330 DATA 0,2,0,0,0,0,5,1,0,0,6,0
340 DATA 0,5,7,0,2,6,0,4,3,0,9,5
350 DATA 4,0,0,0,0,9,0,0,6,0,0,8
360 DIM item$(6),itemname$(6),itempos
(6)
370 FOR i=1 TO 6
380 READ item$(i),itemname$(i),itempo
s(i)
390 NEXT i
400 DATA a lamp,LAMP,5
410 DATA the great diamond,DIAMOND,7
420 DATA a sharp knife,KNIFE,3
430 DATA a hammer,HAMMER,1
440 DATA a mean looking gremlin,GREML
IN,4
450 DATA a nasty little pixie,PIXIE,9
460 DIM com$(7)
470 FOR i=1 TO 7
480 READ com$(i)
490 NEXT
500 DATA GET,TAKE,ON,LIGHT,OFF,DROP,K
ILL
510 DIM direct$(4)
520 FOR i=1 TO 4
530 READ direct$(i)
540 NEXT
550 DATA North,East,South,West
560 DIM bright$(2)
570 bright$(0)="( It's off )"
580 bright$(1)="( It's shining dimly
)"
590 bright$(2)="( It's shining bright
ly )"
600 on=FALSE
610 reallit=2.9
620 lit=2
630 position=1
```

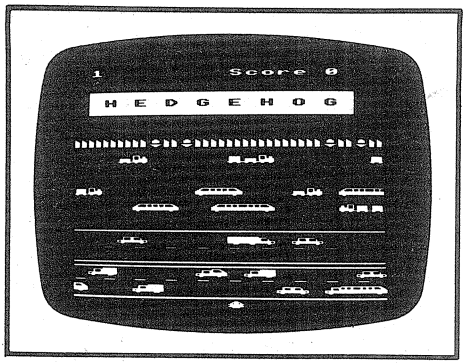
```
  640 dead=FALSE
  650 won=FALSE
  660 moves=0
  670 score=30
  680 carried=0
  690 ENDPROC
  700
  710 DEF PROCtitle
  720 PRINT:PRINT
  730 PRINTCHR$132;"        ************
**************"
  740 PRINTCHR$132;"      ***************
****************"
  750 FORDH=1TO2:PRINTCHR$132;" ****  "
;CHR$133;CHR$141;"Return Of The Diamond
";CHR$140;CHR$132;"  ****":NEXTDH
  760 PRINTCHR$132;"      ***************
****************"
  770 PRINTCHR$132;"        ************
**************"
  780 ENDPROC
  790
  800 DEF PROClook
  810 IF (position=6 OR position=9) AND
   (NOT on OR (itempos(1)<>position AND
itempos(1)<>0)) THEN PRINT"It is pitch
dark.":ENDPROC
  820 PRINT
  830 PRINT"You are ";place$(position)
  840 PRINT
  850 PRINT"Exits :"
  860 FOR i=1 TO 4
  870 IF newpos(position,i)>0 THEN PRIN
T direct$(i);":";
  880 NEXT i
  890 PRINT
  900 PRINT
  910 PRINT"You can see :"
  920 printed=FALSE
  930 FOR i=1 TO 6
  940 IF itempos(i)=position THEN PRINT
 item$(i):printed=TRUE
  950 IF itempos(i)=position AND i=1 AN
D NOT on THEN PRINT bright$(0)
  960 IF itempos(i)=position AND i=1 AN
D on THEN PRINT bright$(lit)
  970 NEXT i
  980 IF NOT printed THEN PRINT"nothing
."
  990 ENDPROC
 1000
 1010 DEF PROCanalise
 1020 IF LEN(c$)=1 THEN IF INSTR("NESW"
,c$)>0 THEN PROCmove:ENDPROC
 1030 IF c$="LOOK" THEN PROClook:ENDPRO
C
 1040 IF LEFT$(c$,3)="INV" THEN PROCinv
entory:ENDPROC
 1050 IF c$="SCORE" THEN PRINT"Your sco
re is ";score;".":ENDPROC
 1060 IF c$="MOVES" THEN PRINT"Moves ma
de : ";moves:ENDPROC
 1070 PROCother_commands
 1080 ENDPROC
 1090
 1100 DEF PROCtime_passing
 1110 score=score-1
 1120 moves=moves+1
 1130 dimmed=FALSE
 1140 IF on THEN reallit=reallit-0.1:di
mmed=TRUE
 1150 lit=INT(reallit)
 1160 IF dimmed AND lit=0 THEN PRINT"Yo
ur lamp just went out.":on=FALSE
 1170 won=(position=8 AND itempos(2)=8)
 1180 ENDPROC
 1190
 1200 DEF PROCmove
 1210 dir=INSTR("NESW",c$)
 1220 IF newpos(position,dir)=0 THEN PR
INT"You can't move in that direction.":
ENDPROC
 1230 IF (position=6 OR position=9) AND
 (NOT on OR (itempos(1)<>position AND i
tempos(1)<>0)) THEN PRINT"You have fall
en into a snake pit!":dead=TRUE:ENDPROC
 1240 position=newpos(position,dir)
 1250 PROClook
 1260 ENDPROC
 1270
 1280 DEF PROCinventory
 1290 PRINT
 1300 PRINT
 1310 PRINT"You are carrying :"
 1320 printed=FALSE
 1330 FOR i=1 TO 6
 1340 IF itempos(i)=0 THEN PRINT item$(
i):printed=TRUE
 1350 IF itempos(i)=0 AND i=1 AND NOT o
n THEN PRINT bright$(0)
 1360 IF itempos(i)=0 AND i=1 AND on TH
EN PRINT bright$(lit)
 1370 NEXT i
 1380 IF NOT printed THEN PRINT"nothing
."
 1390 ENDPROC
 1400
 1410 DEF PROCother_commands
 1420 comno=FNcommand
 1430 thingno=FNthing
 1440 IF comno=0 OR thingno=0 THEN PRIN
T"Sorry. I don't understand.":ENDPROC
 1450 ON comno GOTO 1460,1460,1470,1470
,1480,1490,1500
 1460 PROCtake:ENDPROC
 1470 PROClight:ENDPROC
 1480 PROCoff:ENDPROC
 1490 PROCdrop:ENDPROC
 1500 PROCkill:ENDPROC
 1510 ENDPROC
```

```
1520
1530 DEF FNcommand
1540 no=0:i=0
1550 REPEAT
1560 i=i+1
1570 IF LEFT$(c$,LEN(com$(i)))=com$(i)
THEN no=i
1580 UNTIL no>0 OR i=7
1590 =no
1600
1610 DEF FNthing
1620 no=0:i=0
1630 REPEAT
1640 i=i+1
1650 IF RIGHT$(c$,LEN(itemname$(i)))=i
temname$(i) THEN no=i
1660 UNTIL no>0 OR i=6
1670 =no
1680
1690 DEF PROCtake
1700 IF itempos(thingno)<>position THE
N PRINT"I don't see that here.":ENDPROC
1710 IF thingno=5 OR thingno=6 THEN PR
INT"You'll be lucky!":ENDPROC
1720 IF carried=3 THEN PRINT"You can't
 carry any more.":ENDPROC
1730 itempos(thingno)=0
1740 PRINT"O.K."
1750 carried=carried+1
1760 ENDPROC
1770
1780 DEF PROClight
1790 IF itempos(thingno)<>0 THEN PRINT
"I would if you had it.":ENDPROC
1800 IF thingno<>1 THEN PRINT"You're j
oking!":ENDPROC
1810 IF on THEN PRINT"It's already on.
":ENDPROC
1820 IF lit=0 THEN PRINT"It won't reli
ght.":ENDPROC
1830 PRINT"O.K."
1840 on=TRUE
1850 ENDPROC
1860
1870 DEF PROCoff
1880 IF itempos(thingno)<>0 THEN PRINT
"You're not carrying that.":ENDPROC
1890 IF thingno<>1 THEN PRINT"Come off
 it!":ENDPROC
1900 IF NOT on THEN PRINT"It's already
 off.":ENDPROC
1910 PRINT"O.K."
1920 on=FALSE
1930 ENDPROC
1940
1950 DEF PROCdrop
1960 IF itempos(thingno)<>0 THEN PRINT
"But you haven't got that.":ENDPROC
1970 itempos(thingno)=position
1980 PRINT"O.K."
1990 carried=carried-1
2000 ENDPROC
2010
2020 DEF PROCkill
2030 IF itempos(thingno)<>position THE
N PRINT"I don't see that here.":ENDPROC
2040 IF thingno=5 THEN PROCkill_gremli
n:ENDPROC
2050 IF thingno=6 THEN PROCkill_pixie:
ENDPROC
2060 PRINT"You're joking!"
2070 ENDPROC
2080
2090 DEF PROCkill_gremlin
2100 IF itempos(3)=0 THEN PRINT"You sl
ash your knife at the gremlin and kill
 it easily.":itempos(5)=-1:score=score+1
0:ENDPROC
2110 IF itempos(4)=0 THEN PRINT"You th
row your hammer at the gremlin,but it c
atches it and throws it back.":ENDPROC
2120 PRINT"You fight the gremlin bare
handed,but only succeed in getting kill
ed."
2130 dead=TRUE
2140 ENDPROC
2150
2160 DEF PROCkill_pixie
2170 IF itempos(4)=0 THEN PRINT"You th
row your hammer at the pixie......A hit
!":itempos(6)=-1:score=score+10:ENDPROC
2180 IF itempos(3)=0 THEN PRINT"You sl
ash your knife at the pixie but it dodg
es.":ENDPROC
2190 PRINT"You fight the pixie bare ha
nded but     it is stronger than you th
ought. You get killed."
2200 dead=TRUE
2210 ENDPROC
2220
2230 DEF PROCfinish
2240 PRINT
2250 PRINT
2260 IF won THEN PRINT"      Congratula
tions!!!   You won!!!"
2270 IF dead THEN PRINT"      Bad Luck!
!!   You lost!!!":score=0
2280 PRINT
2290 PRINT
2300 PRINT"    You took ";moves;" moves
."
2310 PRINT"    and your final score was
 ";score;"."
2320 PRINT"    Press SPACE to play agai
n."
2330 REPEAT
2340 UNTIL INKEY$(0)=" "
2350 ENDPROC
```

Program tested on O.S. 0.1 and 1.2

HEDGEHOG (32k)

by Alan Dickinson



Hedgehog is a really first class implementation of the "Frogger" type arcade game. It is fast moving, responsive, the graphics are good, and it makes pretty addictive playing. Essentially you are a hedgehog. You can move left, right or "scurry". Your aim in life is to scurry across a four lane dual carriageway and a busy railway track to gather acorns for supper. It is the rush hour, so the traffic is pretty busy, which means you need concentration and good "scurry" control. If you do get run over by a juggernaut or an Intercity 125, an ambulance will quickly drive to the scene.

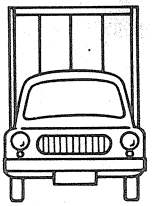
You have three lives, and your score depends on your progress. As your score increases, so the traffic gets steadily worse. Playing tips:

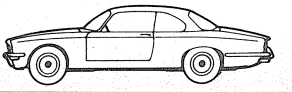
1. Do not roll into a ball in the oncoming traffic.
2. Make good use of the rest point between the road and the railway track.
3. Do not panic at the tooting of horns.

Instructions are given with the program; and as you will see, it is well structured with some well chosen procedure names like PROC-DEFINE, PROC-DRAW-SCREEN, PROC-SPLAT, PROC-CHUCKLE etc. If you want to run this program from disc, you will need to use the "Disc Program Relocator" given elsewhere in this issue.

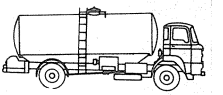
```
  10  REM ************************
  20  REM * PROGRAM. HEDGEHOG
  30  REM * AUTHOR.  ALAN DICKINSON
  40  REM * BEEBUG.
  60  REM ************************
  70  :
  80 ON ERROR MODE7:REPORT:END
  90 :
 100 MODE2
 110 VDU23;8202;0;0;0;
 120 *FX9,25
 130 *FX10,25
 140 DIM R$(9):REM ROADS
 150 DIM A%(19):REM ACORNS
 160 :
 170 PROC_INTRO
 180 PROC_DEFINE
 190 PROC_INIT
 200 REPEAT
 210  PROC_GAME
 220  IF S%>T% PROC_TOPSCORE
 230  PROC_GAMEOVER
 240  TIME=0:REPEAT UNTIL TIME>200
 250  *FX15,1
 260  A$=GET$
 270 UNTIL FALSE
 280 :
 290 DEF PROC_INTRO
 300  COLOUR 133
 310  CLS
 320  COLOUR 15
 330  PRINT''''  H e d g e h o g"
 340  COLOUR 4
 350  PRINTTAB(2,14)"Controls..."
 360  PRINT''"  Z = LEFT"
 370  PRINT'"  X = RIGHT"
 380  PRINT'"  / = SCURRY"
 390  PRINTTAB(2,30)"Snatch acorns...
";
 400  TIME=0:REPEAT UNTIL TIME>300
 410  COLOUR 8
 420  PRINTTAB(2,30)"Press any key...
";
 430  A$=GET$
 440 ENDPROC
 450 :
 460 DEF PROC_DEFINE
 470  VDU23,255,136,204,238,238,238,2
38,238,238
 480  VDU23,254,0,0,-1,-1,0,0,0,0
 490  VDU23,252,0,-1,-1,0,0,-1,-1,0
 500  VDU23,251,0,0,0,0,-1,-1,0,0
 510  VDU23,253,24,60,90,255,126,255,
60,0
 520  VDU23,250,108,72,200,252,254,25
1,127,60
```

```
  530  VDU23,249,24,60,60,60,0,126,60,
24
  540  H$=CHR$17+CHR$1+CHR$253
  550  VDU23,224,15,105,105,249,255,25
5,255,102
  560  VDU23,225,240,150,150,159,255,2
55,255,102
  570  VDU23,226,0,0,0,126,126,126,255
,102
  580  VDU23,227,0,126,126,126,126,126
,255,102
  590  VDU23,228,31,49,97,255,255,255,
255,48
  600  VDU23,229,255,36,36,255,255,255
,255,0
  610  VDU23,230,248,140,134,255,255,2
55,255,24
  620  VDU23,231,0,31,17,17,255,191,25
5,48
  630  VDU23,232,0,240,136,136,255,253
,255,12
  640  VDU23,233,255,255,255,255,255,2
55,255,28
  650  VDU23,234,255,239,199,239,255,2
55,221,28
  660 ENDPROC
  670 :
  680 DEF PROC_INIT
  690  T%=500
  700  T$="Henry Hedgehog"
  702  S1$=" ":S2$="  "
  704  S3$="   ":S4$="    "
  706  S5$="     ":S6$="      "
  710  R$(0)=STRING$(20," ")
  720  R$(1)=S2$+CHR$227+CHR$226+CHR$2
27+CHR$227+CHR$225+S4$+CHR$226+CHR$225
+S5$+CHR$227+CHR$226+CHR$225+S4$
  730  R$(2)=CHR$224+CHR$226+CHR$227+C
HR$226+CHR$226+S5$+CHR$224+CHR$226+S3$
+CHR$224+CHR$227+CHR$227+CHR$226+S2$+C
HR$224+CHR$226+S3$
  740  R$(3)=CHR$228+CHR$229+CHR$229+C
HR$230+S4$+CHR$227+CHR$227+CHR$225+S6$
+CHR$228+CHR$229+CHR$230+S3$+CHR$226+C
HR$225+S1$
  750  R$(4)=CHR$228+CHR$229+CHR$229+C
HR$230+S4$+CHR$224+CHR$227+CHR$227+S6$
+CHR$228+CHR$229+CHR$230+S3$+CHR$224+C
HR$226+S1$
  760  R$(5)=STRING$(20," ")
  770  R$(6)=S2$+CHR$228+CHR$229+CHR$2
29+CHR$232+S5$+CHR$231+CHR$232+S5$+CHR
$233+CHR$233+CHR$232+S1$+CHR$231+CHR$2
32+S2$
  780  R$(7)=CHR$231+CHR$232+S2$+CHR$2
31+CHR$232+S2$+CHR$233+CHR$232+S5$+CHR
$228+CHR$232+S1$+CHR$233+CHR$232+S5$
  790  R$(8)=CHR$231+CHR$232+S1$+CHR$2
31+CHR$232+S3$+CHR$231+CHR$233+S5$+CHR
$231+CHR$230+S1$+CHR$231+CHR$233+S5$
  800  R$(9)=S4$+CHR$231+CHR$232+S1$+C
HR$231+CHR$229+CHR$229+CHR$230+S3$+CHR
$231+CHR$233+S3$+CHR$231+CHR$233+S3$+C
HR$231+CHR$230+S1$
  810  FOR I%=1 TO 9
  820   R$(I%)=STRING$(5,R$(I%))
  830   R$(I%)=LEFT$(R$(I%),100)
  840  NEXT
  850 ENDPROC
  860 :
  870 DEF PROC_GAME
  880  S%=-100:LEVEL%=-1:PROC HARDER
  890  N%=0
  900  REPEAT
  910   N%=N%+1:PROC_GO
  920  UNTIL N%=3
  930  IF S%<1000 ENDPROC
  940  COLOUR 0:COLOUR 129:CLS
  950  PRINTTAB(2,10)"Bonus hedgehog";
  960  PRINTTAB(2,20)"May the fleas";
  970  PRINTTAB(2,22)" be with you";
  980  FOR I%=0 TO 255 STEP1
  990  SOUND&11,-12,I%,2:SOUND&12,-12,
200-I%,2
 1000  NEXT
 1010  *FX15,1
 1020  A$=INKEY$(200)
 1030  REPEAT:N%=N%+1:PROC_GO:UNTIL N%
=4
 1040 ENDPROC
 1050 :
 1060 DEF PROC_GO
 1070  X%=10:Y%=30:Z%=10
 1080  PROC_DRAWSCREEN
 1090  REPEAT
 1100  PROC_ROAD(1,0,12)
 1110  PROC_ROAD(6,0,22)
 1120  PROC_ROAD(7,0,24)
 1130  PROC_ROAD(4,1,18)
 1140  PROC_ROAD(8,1,26)
 1150  PROC_ROAD(9,1,28)
 1160  PROC_ROAD(7,0,24)
 1170  PROC_ROAD(2,1,14)
 1180  PROC_ROAD(8,1,26)
 1190  PROC_ROAD(3,0,16)
 1200  UNTIL Y%=0
 1210  IF ACORNS%=0 PROC_HARDER
 1220 ENDPROC
 1230 :
 1240 DEF PROC_DRAWSCREEN
 1250  COLOUR 0
 1260  COLOUR 128
 1270  CLS
 1280  COLOUR 129
 1290  PRINTTAB(1,4)SPC(17);
 1300  PRINTTAB(1,5)" H E D G E H O G
";
 1310  PRINTTAB(1,6)SPC(17);
 1320  COLOUR3
 1330  COLOUR 128
```





```
 1340  PRINTTAB(1,1);N%;
 1350  PRINTTAB(0,10)STRING$(20,CHR$25
5)
 1360  COLOUR 9
 1370  FOR I%=0 TO 19
 1380  IF A%(I%)=1 PRINTTAB(I%,10)CHR$
249
 1390  NEXT
 1400  COLOUR 7
 1410  PRINTTAB(0,21)STRING$(20,CHR$25
4)
 1420  PRINTTAB(0,23)STRING$(10,"- ")
 1430  PRINTTAB(0,25)STRING$(20,CHR$25
2)
 1440  PRINTTAB(0,27)STRING$(10,"- ");
 1450  PRINTTAB(0,29)STRING$(20,CHR$25
1)
 1460  PRINTTAB(10,1)"Score ";S%;
 1470  COLOUR 12
 1480  PRINTTAB(0,20)R$(5);
 1490  PRINTTAB(X%,Y%)H$;
 1500 ENDPROC
 1510 :
 1520 DEF PROC_HOG
 1530  IF Y%=0 ENDPROC
 1540  IF INKEY-98 AND X%>0 PROC_LEFT
ELSE IF INKEY-67 AND X%<19 PROC_RIGHT
ELSE IF INKEY-105 PROC_FWD
 1550  IFZ%<10 IF MID$(R$(Z%),X%+1,1)<
>S1$ PROC_SPLAT
 1560 ENDPROC
 1570 :
 1580 DEF PROC_LEFT
 1590  X%=X%-1:PRINTTAB(X%,Y%)H$;" ";
 1600 ENDPROC
 1610 :
 1620 DEF PROC_RIGHT
 1630  PRINTTAB(X%,Y%)" ";H$;:X%=X%+1
 1640 ENDPROC
 1650 :
 1660 DEF PROC_FWD
 1670  IF Z%=1 AND A%(X%)<>1 SOUND 1,-
15,40,1:ENDPROC
 1680  Y%=Y%-2
 1690  Z%=Z%-1
 1700  PRINTTAB(X%,Y%)H$;TAB(X%,Y%+2)S
1$;
 1710  SOUND 1,-12,220,1
 1720  SOUND 2,-12,232,1
 1730  S%=S%+10
 1740  PRINTTAB(10,1)"Score ";S%;
 1750  IF Z%=0 PROC_CHUCKLE
 1760 ENDPROC
 1770 :
 1780 DEF PROC_ROAD(R%,D%,SY%)
 1790  IF D%>0 R$(R%)=MID$(R$(R%),2,99
)+LEFT$(R$(R%),1) ELSE R$(R%)=MID$(R$(
R%),100,1)+LEFT$(R$(R%),99)
 1800  COLOUR (R%MOD5)+2
 1810  COLOUR 128
 1820  PRINTTAB(0,SY%)LEFT$(R$(R%),20)
;
 1830  IF SY%=Y% PRINTTAB(X%,Y%)H$;
 1840  PROC_HOG
 1850  IF Y%=SY% IF MID$(R$(R%),X%+1,1
)<>S1$ PROC_SPLAT
 1860  IF RND(80)=80 SOUND3,-15,1,10:S
OUND2,-15,9,8:SOUND1,-15,17,2
 1870 ENDPROC
 1880 :
 1890 DEF PROC_SPLAT
 1900  IF Y%=0 ENDPROC
 1910  COLOUR 14
 1920  PRINTTAB(X%,Y%)"*";
 1930  VDU19,0,1,0,0,0
 1940  COLOUR 12
 1950  COLOUR 139
 1960  PRINTTAB(1,4)SPC(17);
 1970  PRINTTAB(1,5)"    S P L A T
";
 1980  PRINTTAB(1,6)SPC(17);
 1990  COLOUR 128
 2000  FOR I%=1 TO 15
 2010   SOUND 1,-5,255-10*I%,1
 2020   SOUND 0,I%-15,7,2
 2030  NEXT
 2040  COLOUR 1
 2050  PRINTTAB(X%,Y%)CHR$250;
 2060  VDU20
 2070  IF X%>10 AX%=0:AMB$=" "+CHR$234
+CHR$232 ELSE AX%=17:AMB$=CHR$231+CHR$
234+" "
 2080  FOR I%=0 TO 15
 2090  IF AX%<X% PRINTTAB(AX%,Y%)AMB$;
:AX%=AX%+1
 2100  IF AX%>X% AX%=AX%-1:PRINTTAB(AX
%,Y%)AMB$;
 2110  FOR J%=0 TO 120
 2120  SOUND&11,-I%,J%,5
 2130  NEXT
 2140  FOR J%=J% TO 0
 2150  SOUND&11,-I%,J%,5
 2160  NEXT
 2170  NEXT
 2180  Y%=0
 2190 ENDPROC
 2200 :
 2210 DEF PROC_CHUCKLE
 2220  FOR I%=1 TO 25
 2230  FOR J%=1 TO 3
 2240   SOUND J%,-15+I%/3,200-RND(3)*I
%,1
 2250  NEXT
 2260  NEXT
 2270  N%=N%-1:Y%=0:S%=S%+25:A%(X%)=0:
ACORNS%=ACORNS%-1
 2280 ENDPROC
 2290 :
 2300 DEF PROC_HARDER
 2310  FOR I%=1 TO 3
```



```
 2320  FOR J%=30 TO 180 STEP5
 2330   SOUND &11,-12-I%,J%+I%*10,1
 2340   SOUND &12,-11-I%,J%+12+I%*10,1
 2350  NEXT
 2360  NEXT
 2370  S%=S%+100
 2380  FOR I%=0 TO 19:A%(I%)=0:NEXT
 2390  FOR I%=1 TO 4
 2400   REPEAT
 2410    J%=RND(17)+1
 2420   UNTIL A%(J%)=0
 2430   A%(J%)=1
 2440  NEXT
 2450  ACORNS%=4
 2460  LEVEL%=LEVEL%+2
 2470  R$(5)=STRING$(LEVEL%,CHR$255)+S
TRING$(20-LEVEL%*2," ")+STRING$(LEVEL%
,CHR$255)
 2480 ENDPROC
 2490 :
 2500 DEF PROC_TOPSCORE
 2510  COLOUR 3
 2520  COLOUR 129
 2530  CLS
 2540  PRINTTAB(1,3)"T O P - S C O R E
";
 2550  PRINTTAB(1,8)"Enter your name";
 2560  PRINTTAB(1,10);
```

```
 2570  FOR I%=100 TO 200 STEP 12
 2580   SOUND 1,-15,I%,3
 2590   SOUND 2,-14,I%+8,3
 2600  NEXT
 2610  *FX15,1
 2620  INPUT ""T$
 2630  T%=S%
 2640 ENDPROC
 2650 :
 2660 DEF PROC_GAMEOVER
 2670  COLOUR 4
 2680  COLOUR 133
 2690  CLS
 2700  PRINTTAB(5,3)"Super hog";
 2710  COLOUR 15
 2720  PRINTTAB((20-LEN(T$))DIV2,6)T$;
 2730  COLOUR 4
 2740  PRINTTAB(5,11)"Top score";
 2750  COLOUR 0
 2760  PRINTTAB(8,13);T%;
 2770  COLOUR 4
 2780  PRINTTAB(5,18)"Your score";
 2790  COLOUR 0
 2800  PRINTTAB(8,20);S%;
 2810  COLOUR 4
 2820  PRINTTAB(4,30)"Press any key";
 2830 ENDPROC
```

SOFTWARE UPDATE SOFTWARE UPDATE SOFTWARE UPDA

MAGIC EEL JOYSTICKS

Some members have asked how to use joysticks with the Magic Eel game in the BEEBUG software library. To achieve this add the following four lines to the program.

```
181 IF ADVAL(2)>50000 D=-4
182 IF ADVAL(2)<15000 D=4
183 IF ADVAL(1)<15000 D=8
184 IF ADVAL(1)>50000 D=-8
```

This allows you to use the right hand joystick to control your eel. This will save wear and tear on the keyboard, but it does make the eel harder to control.

MASTERFILE UPDATE

Masterfile continues to be one of our most popular titles. Here are a few small changes to bring version 9.0 up to the standard of version 9.1. They should be made to the SECOND program on the tape, not the first.

```
Change line 3020 ..........SPC10
         to 3020 ..........SPC6

amend line 15610 NEXT:record=to
amend line 25230 ......IF to<0 OR to>MR THEN 25230
```

These changes make Masterfile into version 9.1, which should also be altered in the initial REM statement.

BEEBUG NEW ROM OFFER

1.2 OPERATING SYSTEM ROM DEAL
ACORN PRESS RELEASE TO BEEBUG MEMBERS

A special arrangement has been agreed between Acorn and BEEBUG whereby BEEBUG members may obtain the Series One Machine Operating System in ROM at the price of £5.85 including VAT and post and packing.

The ROM will be supplied with fitting instructions to enable members to install it in their machine.

If the computer does not subsequently operate correctly, members may take their machines to an Acorn dealer for the upgrade to be tested, which will be done at a charge of £6.00 plus VAT. This charge will be waived if the ROM is found to have been defective. If the computer has been damaged during the installation process, the dealer will make a repair charge.

NOTES ON ORDERING

1. To get a new ROM, BEEBUG members should send a cheque, made payable to BEEBUG, for £5.85 to: ROM Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD. It is ESSENTIAL to include a cheque with order, and to give your membership number.
2. ROM orders must not be combined with any other order - eg for software etc. Multiple orders can be accepted but there can be no quantity discount in such cases, and the price of £5.85 per unit remains.
3. Because of uncertainties in supply, please allow 4-6 weeks for delivery. We undertake not to cash cheques until the week prior to despatch; and we will provide a monthly account of the supply situation in BEEBUG. Please keep a note of the date on which you posted your order so that you can relate this to future announcements.
4. Please note that we cannot accept EPROM-based operating systems (0.1 or 1.0) in lieu of payment. The exchange of EPROMs for the new operating system can only be performed by Acorn dealers or by Acorn's service centre at Feltham.

ADDRESS: ROM Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD.

WORDWISE Word Processor

BEEBUG Discount 13% SAVE £5

This is a highly sophisticated word processing package for the BBC Micro, and compares favourably with those currently available on other microcomputers. It makes full use of the BBC Micro's advanced facilities, and text is typed and edited in the 40 cloumn Teletext mode, saving memory, thus allowing it to be used with more or less any TV. Wordwise will work equally well on cassette or disc based systems, and it is easy to transfer files from cassette to disc if you upgrade at a later date.

★★★★★★Wordwise now includes a free "TYPING TUTOR" program.★★★★★★

[NOTE: Wordwise requires a series 1 OS.]
The normal price of Wordwise is £39+VAT=£44.85 plus post and packing.
To BEEBUG members the price is £33.88+90p(p&p)+VAT=£40 fully inclusive.
Price to members outside UK is also £40, this includes the extra p&p but NOT VAT.
Cheques MUST be in 'Pounds Sterling'.

EXTRA-SPECIAL OFFER
WORDWISE PLUS 1.2 ROM

Wordwise package plus new 1.2 ROM is offered to members at £45.00 including p&p and VAT.
Price to members outside UK is also £45, this includes the extra p&p but NOT VAT.
Cheques must be in 'Pounds Sterling'.
Cash with order must be sent for BOTH offers. Make cheques payable to "BEEBUG" and send to: Wordwise Offer, PO BOX 50, St Albans, Herts, AL1 2AR.
It is essential to quote your membership number. Please allow 10 days for delivery on the Wordwise offer, and 28 days on the Special Double offer.

IF YOU WRITE TO US

BACK ISSUES (Members only)

All back issues are kept in print (from April 1982). Send 90p per issue PLUS an A5 SAE to the subscriptions address. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that BEEBUG supplements are not supplied with back issues.

Subscriptions Address
BEEBUG
Dept 1
374 Wandsworth Rd
London
SW8 4TE

SUBSCRIPTIONS

Send all applications for membership, subscription renewals, and subscription queries to the subscriptions address.

Membership costs: £5.40 for 6 months (5 issues)
: £9.90 for 1 year (10 issues)
European Membership £16 for 1 year.
Elsewhere - Postal Zone A £19, Zone B £21, Zone C £23

SOFTWARE AND ROM OFFER (Members only)

These are available from the address opposite, which is our NEW software address. (Note that this does not apply to Wordwise - in this instance please see magazine for details).

Software Address
BEEBUG
PO BOX 109
Baker Street
High Wycombe
Bucks
HP11 2TD

IDEAS, HINTS & TIPS, PROGRAMS, AND LONGER ARTICLES

Substantial articles are particularly welcome and we will pay around £25 per page for these, but in this case please give us warning of anything that you intend to write. In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "Minitext Editor" or other means. If you use cassette, please include a backup copy at 300 baud.

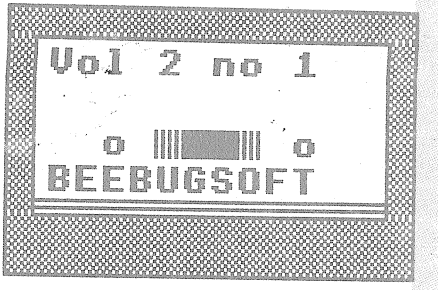
Editorial Address
BEEBUG
PO Box 50
St Albans
Herts
AL1 2AR

We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

BEEBUG NEWSLETTER is edited and produced by Dr David Graham and Sheridan Williams.
Technical Editor: Colin Opie. Production Editor: Phyllida Vanstone.
Technical Assistant: Alan Webster.
Thanks are due to John Yale, Adrian Calcraft, and Tim Powys-Lybbe for assistance with this issue.

MAGAZINE CASSETTE OFFER

To save wear and tear on fingers and brain, we will be offering each month a cassette of the programs featured in the latest edition of BEEBUG. The first program on each tape is a menu program, detailing the tape's contents, and allowing the selection of individual programs. The tapes are produced to a high technical standard by the process used for the BEEBUGSOFT range of titles. Ordering information, and details of currently available cassettes are given below.



Vol 1 no 10
Programs include:

3D Rotation
Life
Artillery Duel
Square Dance
Disc Formatter
File Handler
Micro-Sketch
Date validation Routine

Vol 2 no 1
Programs include:

Four in a Row
Invasion
Music Composer
Bar Chart Generator
Basic Utility Editor
Double Height Routine
Disc Menu Program
PLUS FULL LISTING OF MINI TEXT ED VERSION 3, AND A PROGRAM TO PLAY BEETHOVEN'S BAGATELLE IN b FLAT (3 VOICES)

Vol 2 no 2
Programs include:

Diamond Castle Adventure
Hedgehog
Sound Wizard
Ellipto
Rotating/Expanding Characters
Multiple Function Keys
Compact Machine Code Procedure
PLUS FULL LISTING OF SEIKOSHA SCREEN DUMP

PRICE:Members UK Only £3.50 inc p&p and VAT
Europe £4.50 inc p&p (VAT not charged)
Elsewhere £5.00 inc p&p (VAT not charged)

Please give membership number. Non-Members add 50p per cassette.
If ordering more than one tape, deduct 20p from the price of each.
Make cheques payable to BEEBUG, and post to BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD.

BEEBUG BINDER OFFER

A hard-backed binder for BEEBUG magazine is now available. These binders are dark blue in colour with 'BEEBUG' in gold lettering on the spine. They allow you to use the whole of the first volume of the magazine as a single reference book. Individual issues may be easily added or removed.

Binder price U.K. £3.90 inc p & p, and VAT.
Europe £4.90 inc p & p (VAT not charged)
Elsewhere £5.90 inc p & p (VAT not charged)

Make cheques payable to BEEBUG.
Send to Binder Offer, BEEBUG, PO Box 109, Baker Street, High Wycombe, Bucks, HP11 2TD. Please allow 28 days for delivery on U.K. orders.

Printed in England by Staples Printers St Albans Limited at The Priory Press.
ISSN 0263-7561