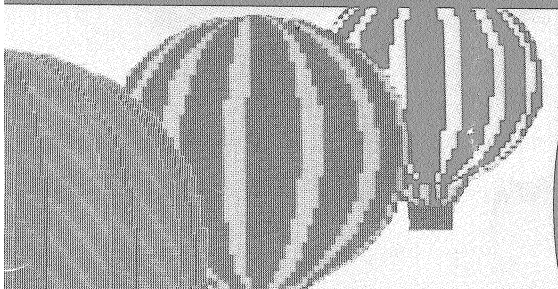
 £1.00

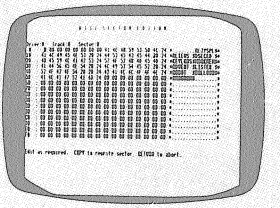
BEEBUG

FOR THE BBC MICRO

Vol 2 No 3 JULY 1983

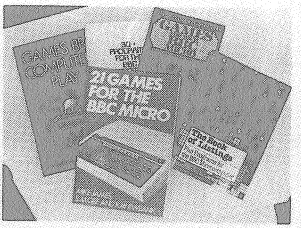


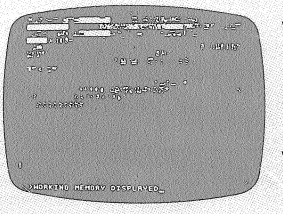
BALLOONS



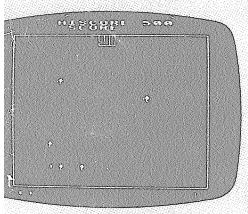
DISC SECTOR EDITOR

FIVE BOOKS OF LISTINGS REVIEWED





WATCHING THE BEEB AT WORK



ROBOT ATTACK

THE NEW EPSON AND SEIKOSHA PRINTERS

ANAGRAMS

BAD PROGRAM LISTER

INTRODUCTION TO DISCS

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

EDITORIAL

MACHINE SUPPLY

Sales of the Beeb have now reached the 120,000 mark, with a current turn-over of around 15,000 per month. It is expected that this will reach 19,000 by July this year. All in all BBC Enterprises must be very pleased with the 10% royalty that they get on each machine sold (which everyone has kept very quiet about). On our calculations this comes to something over £4M.

With the great popularity of the Beeb, what exactly are the turn-around times on orders? Acorn inform us that stocks of model As are reasonable, model Bs are 'tight' - though a 28 day despatch (not delivery) is quoted. Disc machines and Econets, however present a problem. The primary schools' education project is absorbing most of the former, and all of the latter at the time of writing. Econets are expected to be available for other users from the beginning of August, but there will inevitably be a backlog to clear. Some disc machines are being despatched to non-education users, but this is disappointingly few - only a few hundred a month. A similar number of disc upgrade kits are also apparently being despatched to dealers. Demand for the disc interface is high, so the backlog is increasing, though Acorn say that they hope to have turn-around down to about 28 days in the next couple of months. The real problem here has been poor supplies of the 8271 disc controller chip. Acorn are one of the few users of this Intel chip, and although they have increased their orders for it, there is a lead time of several months. In the next issue we hope to be able to announce two unofficial disc upgrades for the Beeb, one of which does away with the need for the 8271.

David Graham

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOA

BEEBMAZE

The excellent program BEEBMAZE published in BEEBUG Vol 1 No 9 was attributed to R.Hull. We have since discovered that the original algorithm and generalised program on which the published version was based were developed by Wynford James in an article in Practical Computing. Our apologies for not including him in the credits.

DISC MASTERFILE

Many members have requested a disc version of MASTERFILE. We expect this to be available from mid August, and will announce details in the next issue.

MAGAZINE CASSETTE

When we first introduced the magazine cassette, we said that we would try to include on them items that we couldn't get into the magazine itself. This month's cassette contains five such items:

A copy of ALIENS (BEEBUG Vol 1 No 6) modified for joysticks.
A copy of RESCUE (BEEBUG Vol 1 No 8) which restores Bad Programs.
A copy of 3D ROTATION (BEEBUG Vol 1 No 10) extended to translate objects.
A machine code screen dump for the Seikosha 250X (instructions on-screen)
An audio reproduction of the full vocabulary of speech PHROM A.

TESTED PROGRAMS

Unless otherwise stated, all programs in this magazine have been tested on operating systems 0.1 and 1.2, and on both Basic I and Basic II. All listings are produced directly from working programs.

HINT WINNERS

This month's hint winners are P.Jollyman (£10), Malcolm Davis (£5), Richard Porter (£5) and Steven Probyn (£5).

BEEBUG MAGAZINE

GENERAL CONTENTS

PROGRAMS

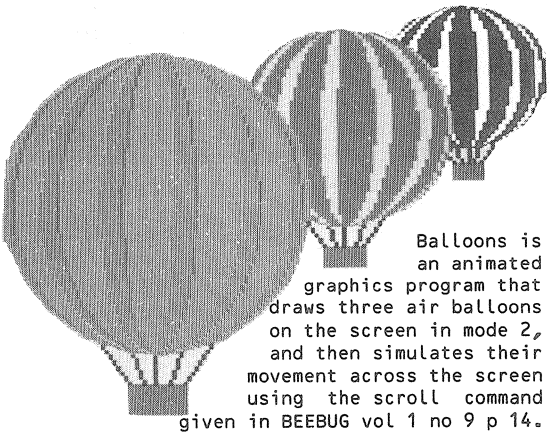
HINTS, TIPS & INFO

ADDITIONAL ITEMS ON MAGAZINE CASSETTE

- Audio Reproduction of Full Vocabulary of Speech PHROM A
- Rescue - A program to recover from "Bad program" message
- 3-D Rotation - Full Listing
- Aliens With Joysticks - Full Listing

BALLOONS (16/32k)

by A. Collyer



Balloons is an animated graphics program that draws three air balloons on the screen in mode 2, and then simulates their movement across the screen using the scroll command given in BEEBUG vol 1 no 9 p 14. At the same time the balloons appear to rotate. This is achieved by the clever use of the VDU 19 statements in lines 170 and 190. To run it on a 16k machine change the 8 and 4 in lines 90 and 100 respectively with 1, change 2 to 5 in line 60 and replace lines 170 and 190 with:

```
170 VDU19,c%,7;0;
190 VDU19,c%,0;0;
```

```
  10 REM Balloons  VERSION  1A
  20 REM Animated Graphics Demonstrati
on
  30 REM A.M.Collyer - 27/3/83
  40 REM BBC Model B (32k Model A)
  50 ON ERROR GOTO 490
  60 MODE2:VDU19,0,6;0;23;11,255;0;0;0
:REM Cursor off
  70 REM Draw three balloons
  80 PROCballoon(1000,800,150,1,-1)
  90 PROCballoon(750,700,200,8,1)
 100 PROCballoon(350,550,300,4,-1)
 110 hpos=0
 120 REM Animated sequence
 130 REPEAT
 140 FORc%=1 TO 3
 150 hpos=(hpos+1)MOD80:VDU23;13,hpos;
0;0;0;:REM Scroll screen left using CRT
C register 13
 160 REM Rotation effect
 170 VDU19,c%,7;0;19,7+c%,3;0;19,3+c%,
2;0;
 180 dummy=INKEY(7):REM Delay colour c
hange
 190 VDU19,c%,0;0;19,7+c%,1;0;19,3+c%,
4;0;
 200 NEXTc%
 210 UNTIL FALSE:REM Go on for ever
 220
 230 REM Procedure to draw ballon
 240 DEFPROCballoon(xpos%,ypos%,size%,
colour%,dir%)
 250 LOCAL angle,col%,x
 260 VDU29,xpos%;ypos%;:PROCbasket(siz
e%)
 270 MOVE0,size%:FORangle=0 TO PI*2 S
TEP 0.3
 280 col%=1+((1+angle/(2*PI))MOD3)
 290 IFdir%*SIN(angle)<0 GCOL0,colour%
+3-col% ELSE GCOL0,col%+colour%-1
 300 x=size%*SIN(angle)*COS(angle/40)*
1.1
 310 PLOT85,x,size%*COS(angle):PLOT85,
x,0
 320 NEXTangle
 330 ENDPROC
 340
 350 REM Procedure to draw basket unde
r balloon
 360 DEFPROCbasket(size%)
 370 LOCAL x,y,I
 380 VDU19,14,1;0;19,15,0;0;
 390 GCOL0,14
 400 x=0.2*size%:y=x
 410 MOVE-x,-y-size%:MOVE-x,-2*y-size%
 420 PLOT85,x,-y-size%:PLOT85,x,-y*2-s
ize%
 430 GCOL0,15
 440 FORI=-x TO x STEPx/2
 450 MOVEI,-y-size%:DRAWI+I*4.4,0
 460 NEXTI
 470 ENDPROC
 480
 490 ON ERROR OFF
 500 MODE 7
 510 IF ERR<>17 REPORT:PRINT" at line
";ERL
 520 END
```

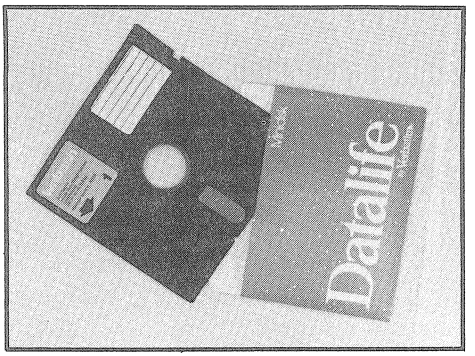
HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

EASIER ENTRY OF LOWER CASE VARIABLE NAMES (New O.S.) - K. Walker, J. Hewes and many others

Pressing SHIFT and CAPS-LOCK together selects a keyboard mode equivalent to normal CAPS-LOCK except that subsequently pressing SHIFT gives lower case letters, thus making entry of lower case variable names easier (though putting them in a function key is easier still). To return to normal mode press CAPS-LOCK twice.

AN INTRODUCTION TO DISCS

by Sheridan Williams



The concept of a disc is not hard to understand. It is merely a disc of plastic or metal covered on both sides by a magnetisable material very similar to the coating on magnetic tape. There are two major types - hard discs, and soft (floppy discs). For this article I will ignore the hard disc and its asscciated drive, as even the cheapest is in excess of £1,500, but the principles of data organisation are very similar to floppies. The disc drives currently available for the BBC micro take 5.25" diameter discs. The other main size is 8", though 3.5" discs (micro-floppies) are under development.

THE ADVANTAGES OF DISCS

The most obvious advantage of discs for the BBC user with only a cassette interface is the great increase in speed achieved when loading and saving programs. A 20k program will save or load in about 2 seconds, though data files take considerably longer.

A less obvious advantage is the provision of the so-called Disc Filing System which provides a whole series of new commands, as well as organising data management. For example typing *.<return> will give an instant catalogue of all titles stored on a given drive. There are also commands to copy files (data or program), to lock them (to prevent accidental erasure), to allow text to be directly read to the screen from a file (useful for program instructions), and a procedure for so called auto-booting a file - this causes a program to be automatically chained in if Shift and Break are pressed simultaneously. This is by no means an exclusive list, (for more details see BEEBUG Vol 1 nos 9 and 10), but it does serve to illustrate some of the advantages of a disc system. The chief drawback to procuring discs is cost - a single drive plus interface will cost around £250 (around £350 if you buy an Acorn drive). There is a secondary problem of memory loss - since a disc interface absorbs an extra 2816 bytes of memory, for work space. You can offset this somewhat by using techniques described in recent issues of BEEBUG. Even with this drawback, and the rather annoying fact that the Beeb will only allow seven-character file names, discs are a great improvement over cassettes - though if you have bought quantities of protected cassette-based programs, you may have trouble getting these on to disc.

TRACKS SECTORS

Data is recorded on the disc in tracks and sectors. The most common numbers of tracks are 35,40,77 or 80. In the BBC micro most systems will be 40 or 80 tracks. These tracks are a series of concentric circles. As the recording surface is only 1 inch in width this gives the width of the track as 1/80 inch. This requires a very precise "stepper" motor to enable the disc head to be positioned quickly and accurately over any track. The reason why there is only a 1 inch wide recording surface is that the outermost portion of the disc is unusable because of warping, and the innermost portion unusable because the recording density would be too high and also have a tendency to warp. Note that the same amount of data is recorded on the outermost track as the innermost even though the length (circumference) of the innermost track is less than half the outermost track.

Each track is divided into sectors. It is the sector which is the

fundamental unit for disc transfers. Whatever is requested to be read or written to the disc, it is always a sector of data that is moved by the BBC DFS. On the BBC there are 10 sectors per track, and each sector holds 256 bytes of data, so that a 40 track disc can hold 40x10x256 bytes - or 100k bytes.

Floppy discs can be either soft or hard sectored. Most BBC drives currently available use soft-sectored discs. This means that the sector markers on the disc are put there magnetically rather than physically. You can tell whether a disc is hard or soft sectored by looking through the hole adjacent to the centre, rotate the drive one complete revolution: if just one hole passes by, the disc is soft sectored, if several pass by, then it is hard sectored. The soft sectored discs used on the Beeb need to be "formatted" before use. This involves placing a new disc into the drive, and running a formatter program which magnetically marks the beginning of each sector. A 40 track disc can be formatted in less than half a minute (see BEEBUG vol 1 no 10 p 25 for a disc formatter).

As I said earlier all discs are coated with magnetisable material on both sides but a quality control process ascertains whether the disc is to be sold as single or double sided. This is one reason why it is unwise to use discs marked "single sided" as "double sided" discs. They will probably appear to work but time will tell and you'll be disappointed when you eventually lose your programs and data on the suspect side. The categorization of a disc into SS/DS, SD/DD (single or double density), 40/77/80 tracks is determined by quality control and hence you may find that any disc (of the correct size) will work in your drive. You should, however, use the disc that fits the specification of your drive unit.

INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO

*CODE AND *LINE Steven Probyn

There are two extra commands in O.S. 1.2, *CODE and *LINE, undocumented in the User Guide. Both are intended for the easy access of user supplied machine code routines. Both of these words indirect through the user vector (USERV) stored at &200,&201, which defaults to point at the Bad Command message and can be redirected by the user.

*CODE X,Y
Jumps with USERV to your routine with A=0 and X and Y set to the parameters.

*LINE FRED
Jumps with A=1 and X,Y containing the address of the string. X low byte, Y high byte, these will point to a location in the input buffer or to the middle of a program. The string is terminated by carriage return (ASCII 13). This should of course be read with OSWORD 5 (read I/O processor memory).

These commands are an easy route to adding new commands to the system for the machine code user.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CURSOR ON/OFF IN ALL MODES (New O.S.) - G.M. Abernethy

In the series 1 operating systems
VDU 23,1,0;0;0;0; turns the cursor off
VDU 23,1,1;0;0;0; turns the cursor on

This is documented on page 77 of the User Guide, but is not mentioned in the reference section of the Guide. Note that pressing any of the cursor control keys reinstates the cursor (this does not happen with VDU 23;11,255;0;0;0;). In some situations this will be an advantage, but in others it may be inconvenient.

GAMES FROM SUPERIOR SOFTWARE AND BUG-BYTE

Reviewed by Alan Webster

Program: Chess
Supplier: Bug-Byte
Price: £8.00
Value: *

Bug-Byte's Chess is quite an extraordinary piece of software in that it appears not to work on either operating system. The game produced a corrupt display on 1.2, and on 0.1 the display was all right, but the game would not allow you to play. I hope that in future Bug-Byte will not sell software that does not run on ANY Operating System, as this could put some people off forever.

In fact we have received several complaints from members regarding Bug-Byte's software. We have asked them to comment, and will report what they say should they reply.

Program: Road Runner
Supplier: Superior Software
Price: £6.50
Value: ****

This is one of the best games available from Superior Software. It is a variation of the arcade game Rally-X in which you steer a car around a maze, dodging 3 enemy cars and boulders in order to catch flags. The game becomes very hard and has very smooth graphics. Excellent.

Program: Airlift
Supplier: Bug-Byte
Price: £5.50
Value: *

Airlift is a game where you navigate a helicopter and you have to pick up 'things' and return them to base. The game runs on a 0.1 but the game was not very exciting, and the graphics left much room for improvement.

Although the game is cheap compared to other software this does not compensate for the low quality, and again this game does not run on the new Operating System.

Program: Space Fighter
Supplier: Superior Software
Price: £6.50
Value: ***

Space Fighter is a cross between 'Defender' and 'Missile Command', but is not as exciting as either of these games. The game itself has good graphics and is in machine code giving decent speeds, the fastest of which is pretty near impossible.

The object of the game is to pilot your spaceship and shoot the attacking aliens that advance towards you. You are also given smart bombs that destroy all of the evil things on the screen, and a certain amount of fuel which diminishes quickly. Fuel dumps run along the bottom of the planet and these have to be destroyed to top up your fuel supply.

There is a high score table and six different speeds which is a nice addition to the program.

Program: Frogger
Supplier: Superior Software
Price: £6.50
Value: ****

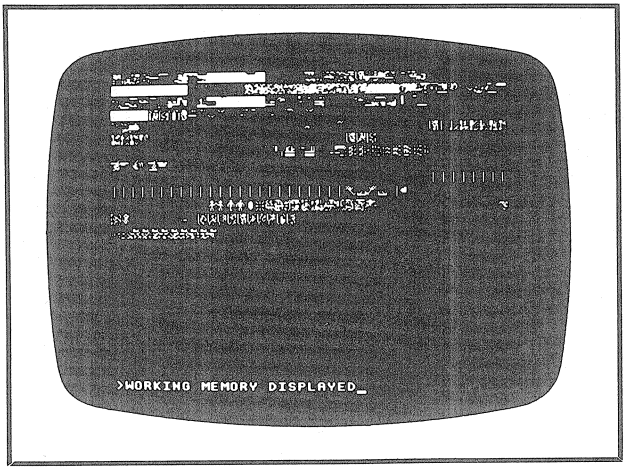
This is one of the many 'Frogger' games appearing on the market at the moment, and it is very good indeed. The aim of the game is to guide your frog across a busy road and a river to the far side of the river bank. A difficult game with fast flicker-free graphics and a frog that really hops!

Program: Space Pirates
Supplier: Bug-Byte
Price: £8.00
Value: **

Space Pirates is loosely based on the arcade game 'Asteroids' with you as the pilot of a space ship trying to prevent the pirates stealing all of your life pods. The game was quite original in theme and was written to run on a 16k machine. This was the best of the Bug-Byte software reviewed.

WATCHING THE BEEB AT WORK (16k)

by Graham Greatrix



How would you like to see the memory in your micro actually carrying out its various tasks? If you would, then clear your machine completely by switching on (you may have to switch off first), and then type in the following. If you have a disc interface, type *TAPE <return> followed by PAGE=&E00 <return>.

```
10 *TV254
20 MODE6
30 VDU23;12;0;0;0;23;
   13;0;0;0;0;
40 ?&34E=0
50 VDU12,28,0,24,39,13,12
60 FORI%=TOP TO TOP+&1200
   :?I%=0:NEXT
```

Now run the program. The lower half of the screen contains a text window for BASIC and the upper half contains a moving picture of the memory from address &0000 to &1000. The diagram shows how this picture of memory is organised.

The 'v' symbols at the top of the table are at intervals of &40. The colons show the boundaries of each named section of memory. 'R-T' is the Run-Time keyboard buffer, 'C' is the Copy area and 'Q' is the sound queue area. There are forty character slots across the screen and each slot represents eight bytes. The first byte is at the top of the slot and the eighth byte at the bottom. The next slot contains the next eight bytes. A white line across the slot represents a byte equal to 255. A Basic program always ends with such a byte. Look at the end of the area labelled "program" where there should be a white line.

KEY PRESSING

Find the position of the Keyboard Buffer. Watch this area while you keep key '8' pressed. As the '8's are printed to the screen, the area fills up with a series of vertical white bands. When the buffer is full, the computer will sound a beep. When it does so, press RETURN. The ASCII code for the number '8' is &38 or 00111000 binary. Thus each byte consists of two unlit pixels, followed by three lit pixels and ending with three unlit pixels. What happens when you press key '3' continuously? Does this pattern correspond to the ASCII for the number '3'? (00110011). Now try a few other keys. Watch the area of the Run-Time buffer. The same patterns appear here as well. However, if you type a control character, such as 'control-C' the Run-Time buffer will respond but the key-board buffer will not. Now try deleting characters from the screen.

COPY

Type LIST and use the edit keys to copy line 60. Watch the memory section marked 'C'.The characters are also inserted into the keyboard buffer.

FUNCTION KEYS

Define function key 1 as a long string of 'A's. Watch the function key area as you press Return. Do the same with function keys 2, 3, etc until this area of memory is full. See the 'Bad key' message appear.

INTEGERS

To make the Integers section spring to life, type the following in immediate mode and watch that section when you press Return:

```
@%=7:A%=7:B%=7:C%=7:D%=7:
E%=7:F%=7:X%=7:Y%=7:Z%=7
```

FLOATING POINT VARIABLES

Typing:

```
A=1:B=1:C=1:DX=1:EX=1:FX=1:
xx=1:yx=1:zx=1
```

will start up the variable names section when you hit Return. Information about these variables is stored at positions that depend on the ASCII value of the first letter of the variable name. Typing CLEAR will clear this section.

CHARACTER DEFINITIONS

The character definition section can be started with

```
VDU23,64,1,1,1,1,1,1,1,1
```

Note how the whole of this section is filled with characters, even though only '@' has been redefined. Now type

```
VDU23,230,1,1,1,1,1,1,1,1
```

Look at the letter 'F' in the row of characters and in a LISTing of line 60 of your program. Type *FX20,0 to recover the situation. Note that if you run this memory display program after playing ROBOTS you will see the defined characters reproduced (see screen photo).

PRINTER BUFFER

The Printer Buffer is activated by Control/B and then LISTing your program to your printer.

STRING VARIABLES

Inserting

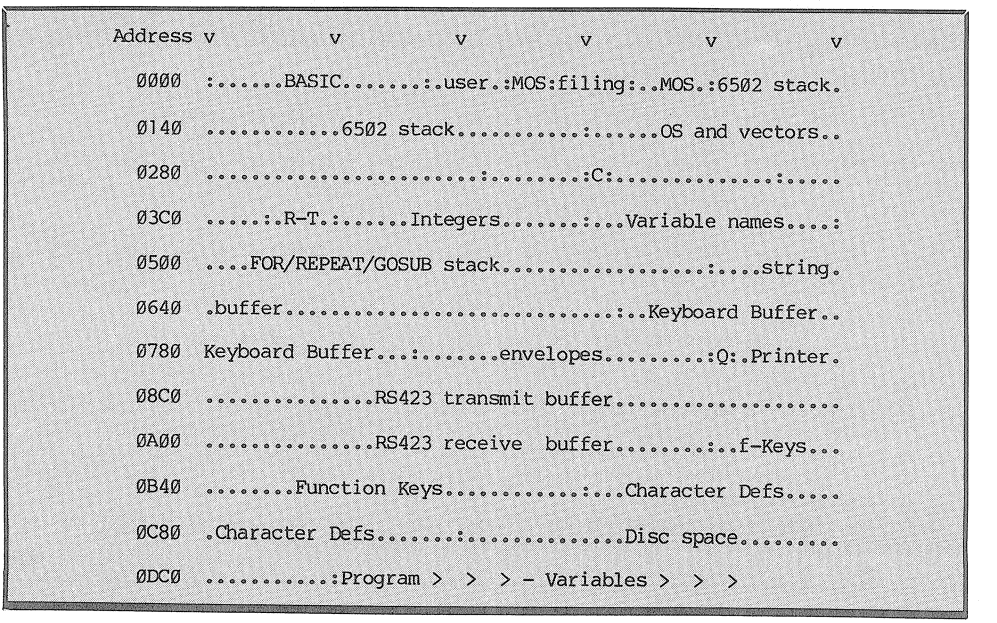
W$="33333333333333333333" will start the string buffer on hitting return.

BASIC

Add the following to your BASIC program.

```
1000 FOR I=1 TO 10000
:X=SQR(I):NEXT
```

On hitting RETURN, watch the section which lies in the clear area following the program.You will be able to see the line being added to the end of the program and a new 255 byte appear at the end of the added line. type GOTO1000 in immediate mode and watch the values of I and X being changed and stored just after the BASIC program. When you hit return, you will also see the FOR/REPEAT/GOSUB stack alter.



ARRAYS

Try adding the following program line:

```
2000 DIM A(50):FOR J=1
     TO 50:A(J)=127:NEXT
```

This will show how an array is stored on typing GOTO 2000.

SOUND BUFFER

Type CTRL-G a number of times. Note the sounds queueing up and the envelopes being activated.

UNUSED

Some sections of memory appear not to be used. For example, the protected zero page locations from &70 to &8F and the area just below PAGE from &D00 to &DFF where machine code routines can be placed.

We have only examined some of the sections of the memory display. There is a lot still to discover, so keep experimenting.

POINTS ARISING

PROGRAM COMPACTER

Two members have noted expressions which should not be compacted (program compacter BEEBUG vol. 1 no. 8 p. 11). The problems occur when taking vital spaces out of statements. These are as follows:

When *SAVE "Screen" 3500 8000 is compacted the space between the two locations is taken out and leaves

```
*SAVE"Screen"35008000
```

The second problem occurs when a variable turns into a number, for instance:

```
IF B=2 E2=1  becomes
IFB=2E2=1
```

which is now wrong because the computer thinks that 2E2 is 200!

Thanks to D Fowles and D Knowles for pointing these out.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

YET MORE LIVES!! - Dave Tyler

Dave Tyler, on reading about 'More Lives in Rocket-Raid' (BEEBUG v.1 no.10 p.34), decided to perform a similar operation on other games. He writes:
I've achieved 'more lives' in 4 other games. These are Snapper, Planetoid, Monsters and Swoop (Program Power). The memory locations and methods are given below. I certainly had fun seeing what screen 13 looks like in Snapper, and 8 in Planetoid (neither of which I am likely to reach with just 3 lives!).

1) Snapper - CHAIN "SNAPPER" as usual but when 'Snap2' has finished loading, press Escape and enter the line:

```
45 ?&FDD=&7F
```

and then 'RUN'.

2) Planetoid - CHAIN "PLANETOID". When 'Planet1' has loaded enter:

```
150 ?&276B=&7F
```

and then 'RUN'.

3) Monsters - LOAD "MONSTERS". 'RUN' but press Escape before anything starts loading. Enter *L."Monsters". When loaded enter ?&1EE5=&7F and then CALL &E02

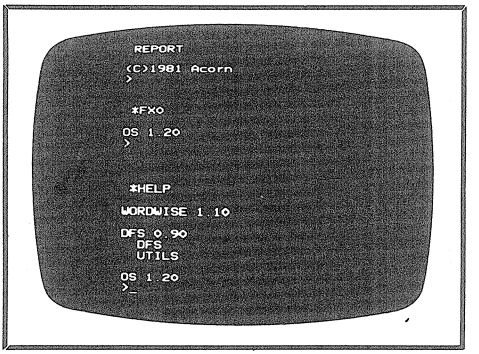
4) Swoop - *L."SWOOP". When loaded enter ?&24E1=&7F and then CALL &2800. This can do strange things to the screen, if it doesn't enter mode 2 try pressing Escape and trying again.

[Note we have not tested these suggestions, and in any case they may not necessarily work with ALL versions of the given programs. Ed]

WHICH VERSION?

by David Graham

Does your machine have an old or a new operating system? Does it have Basic I or Basic II? Which version of the Disc Filing System does it have? David Graham tells you how to find out.



As you are probably aware, most of the systems firmware (ie chip based programs) in the BBC micro, has undergone revision since the first machines were despatched well over a year ago. Although care has been taken to ensure so-called "downward compatibility", it is often important to know exactly what version of the various pieces of firmware you have. For example, some programs commercially available will only work on the old operating system. Here are the details.

OPERATING SYSTEM Ø.1, 1.Ø or 1.2?

To discover which, type

*FXØ <return>

and you will be told which you have - but note, if it is Ø.1, it will say "EPROM Ø.1" whether you have a ROM or EPROM based version. The EPROM version will be exchanged free by Acorn dealers for a 1.2. The 1.2 is the new ROM-based operating system, and 1.Ø is an EPROM based version of the new operating system fitted to early disc machines. For notes on 1.2 see BEEBUG Vol. 1 no. 1Ø p 15 and Vol. 2 no. 2 pp 28-31.

BASIC I or BASIC II?

To discover which, turn on your machine, and type

REPORT <return>

The machine will print a copyright message. If the year on the message is 1981, you have Basic I; if it is 1982, it is Basic II. There are only a few differences between the two chips, see BEEBUG vol. 1 no. 6 p. 11.

DISC FILING SYSTEM - version Ø.9

To discover which, type

*HELP <return>

A number of things will be printed in response to this, including the letters "DFS" followed by the version number. To date all production models of the Disc Filing System have been version Ø.9, though there are some unofficial versions around with higher numbers. We suspect that Acorn are preparing a new version of the DFS for general release, but this is not yet available.

OTHER INFORMATION

The *HELP command (only available on machines with a series 1 Operating System - i.e 1.Ø or 1.2) gives other useful information. For example, as well as printing the operating system number, it prints the name and version number of any other ROMs or EPROMs found in the "paged ROM" sockets - eg "WORDWISE 1.1Ø" etc. This can be useful to know if you are using a machine that is not your own, and want to find out exactly what firmware it has on board.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CLOSING ALL FILES - Matthew Rapier

When using the Disc filing system with text files it is possible for a file to be left open when an error occurs. The file cannot be deleted as 'File open' is produced. The command CLOSE #Ø will automatically close all files and solve the problem.

FIVE BOOKS OF LISTINGS FOR THE BBC MICRO

Reviewed by Alan Webster

Book: Games for your BBC Micro (128 pages)
By: Alex Gollner
Pub: Virgin Books Ltd
Price: £2.95
Value: ***

This is one of the better games books around, not because the games are better - far from it, but because the programming techniques are much better. The games themselves are all standard implementations of typical games (nothing very exciting), and although a couple of programs did not seem to work, the low price is enticing.

Book: The Book of Listings (159 pages)
By: Tim Hartnell & Jeremy Ruston
Pub: BBC Soft
Price: £3.75
Value: *

In my personal view, this book does not warrant the title 'THE' book of listings, there is nothing difinitive about it, it's more 'A' book of listings.

Most of the listings have no spaces in them which makes them quite difficult to read (someone forgot to use LISTO1). There is also a conspicuous absence of procedures, and it looks as though messers Hartnell and Ruston don't know that the BEEB has a useful envelope command!

This book from the BBC sets a poor example both of programming techniques, and in games quality. Its one merit is that it is inexpensive. It is a great pity that the BBC, renowned in other fields, are accepting and publishing such material. It contrasts sharply to Acornsoft's book Creative Graphics reviewed in BEEBUG vol.1 no.9.

Book: Games BBC Computers Play (113 pages)
By: T.Hartnell, S.Gee and M.James
Pub: Interface Publications
Price: £6.95
Value: *

This is another book which in my view is extremely poor value for money. The cover is the best part of this book, and the price is extremely high for a small number of pages. The programs do not make full use of the BBC's facilities, and I suspect that they have been converted from a Sinclair computer with very little modification indeed. This is certainly a book that I would not buy.

Book: 21 Games for the BBC Micro (144 pages)
By: M.James, S.Gee and K.Ewbank
Pub: Granada Publishing Ltd
Price: £5.95
Value: ****

This book is nicely produced and in many respects maintains the high standard set by Granada Publishing for books about the BBC micro. The quality of programming, though somewhat variable, is generally higher than that found in the other four books reviewed here. The games are quite novel with a mixture of the old favourites, and each listing is clearly set out, which makes them easy to type in.

»

Book: 30+ Programs for the BBC Microcomputer
(71 pages)
By: C.Evans
Pub: C.J.E. Microcomputers
Price: £4.95
Value: **

This book consists of 31 programs for the BBC Micro. The programs are put into categories, among them being games, music and utilities.

The games are all in the Teletext mode and therefore fall down on expectation from a visual point of view. The scientific and educational programs are better, and there is a music program which plays God Save the Queen. Although you do get a reference card and a 'Hints and Tips' supplement, the book is still over-priced in my view, and is not produced to a particularly high standard.

MODIFICATIONS TO ACORNSOFT'S "PLANETOID"

by Jeffrey Siddle

These modifications work on Planetoid, they do not work on Defender (Planetoid has a high score table and Defender does not). To use them it is necessary to incorporate the changes in the second header program after the machine code has been loaded. You will have to make a new copy of the game with either a heavily modified second program, or a modified version of the code.

Firstly to change the keys assigned to various functions, put the negative INKEY number of the new key required into the location with ?&address = key number. Negative INKEY values are given on the BEEBUG reference card p.5.

Address	Key checked	Address	Key checked
13E0	Fire Key	1E80	Move down
1A22	Smart Bomb	1ED0	Thrust
1E6C	Move up	1E94	Reverse

Address	Effect
1AF2	Number of lives lost when ship is hit
1814	Number of smart bombs lost when one is used
2638	Initial number of lives and smart bombs
2657	Number of extra lives every 10,000 points
2660	Number of extra smart bombs every 10,000 points

Note: These modifications have been checked at BEEBUG, but there may be more than one version of PLANETOID, and the locations used may differ in different versions.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

INVISIBLE FUNCTION KEYS - G.M. Abernethy

When a function key is pressed the text for it is printed out on the screen. In some cases this is undesirable. To remove the text, define the function key to include cursor up characters and spaces, so as to remove the unwanted parts, e.g.:

```
*KEY8|M|MDIMP&-1:P.CHR$11;CHR$11;CHR$11;H.-P%;"bytes free";SPC90|M
```

The cursor up characters can be embedded with |K as usual for control characters. A little experimentation will be necessary to sort out the correct number of cursor control characters. This hint is most useful for disc users or those disciplined enough to load their function key definitions every time they start programming, as there is considerable extra effort in typing the definition with the extra cursor control characters.

MAKE YOUR MICRO SPEAK LIKE KENNETH KENDALL

BBC Micro Speech Upgrade Reviewed by David Graham.

The speech upgrade for the Beeb will be available shortly from Dealers. The upgrade will cost £55 including VAT, and will need to be performed by a dealer. The reason for this is that some machines require modifications to be pcb to implement speech, and in any case, the upgrade will include the fitting of a cartridge ROM socket in the slot to the left of the keyboard. The ROM socket will then be available for plug-in ROM programs, and for vocabulary expansion of the speech unit.

The speech unit itself essentially requires the addition of two ICs. One is a Texas Instruments speech processor, the other a vocabulary ROM. The ROM supplied with the upgrade (coded PHROM A - (Phrase ROM)) contains a vocabulary of around 160 words or sounds. These include the ASCII set with additions like "hundred", "thousand" etc, and extensions which allow the creation of words such as "seven-ty", "eight-ty" etc.

Speech is accessed very simply using a variant of the SOUND command. For example SOUND -1, ASC"P",0,0 will cause the letter "P" to be spoken. The two zeros at the end are redundant, and are only included to keep the syntax the same as for the normal use of SOUND. The operating system allows words to be called up from the PHROM in different ways. You can define the word required by using the so-called word number, or by using an ASCII letter, or by giving an absolute address in PHROM of the start of the word. This allows great flexibility.

As an example of the use of a word number, the command SOUND -1,160,0,0 will produce the word "Acorn". The speech system user guide contains a directory of words and sounds with their appropriate word numbers and addresses. Sounds with word numbers in the range 32 to 126 have a direct or indirect association with the ASCII characters of the same range, and can be called as in the first example. Thus SOUND -1,ASC"Q",0,0 will produce the SOUND "Q". The pound sign produces the word "pound", and so on. These may be easily accessed with the following simple program given in the speech system user guide:

```
REPEAT: SOUND -1,GET,0,0:UNTIL 0
```

This will "speak" any key pressed, with the lower case letters producing a range of complete words. The following program will speak the whole vocabulary of PHROM A. This takes about two minutes, suggesting how well packed the data is. Traditionally stored data would require a 500k byte ROM to generate 2 minutes of speech.

```
10 FOR A% = 32 TO 291
20 SOUND -1, A%,0,0
30 NEXT
```

More words can be generated by combining sounds - for example EIGHT + TEEN (there is a special shortened version of the numbers for this purpose) and COMPUTE + ING etc.

Words may also be called up by a direct address, and a particular advantage of this is that the data for the word can be in RAM. This enables the user to experiment with the sounds produced. And although the data blocks used are so complex as to preclude the creation of new words, it would certainly be possible to edit data for words already supplied. It is also possible to access speech from machine code using special OSBYTE and OSWORD calls. This is documented in the clearly written manual.

As I hope I have implied, using the system is extremely easy, moreover the quality of reproduction is extremely good. There are just three keyboard letters that do not sound quite right (B, D and M), but for the rest, the words are not only understandable, but clearly articulated in an obviously English accent which still retains something of its maker Kenneth Kendall. (It was Kenneth Kendall whose voice was used to create the data for PHROM A.)

The speech upgrade provides a most interesting addition to the BBC micro's

facilities, and compares very favourably in quality with add-on speech units for other micros. Its most obvious immediate use would appear to be for the visually handicapped, and in primary education, where programs of the 'speak and spell' variety could be used as a teaching aid in spelling and simple arithmetic. One other area of application could be in permitting computer output when the operator cannot see the screen, or when the screen is in full use, as for example in 'speaking cursor co-ordinates' in a screen drawing program. It could also enhance strategy games such as chess or Star Trek; though PHROM A vocabulary could prove somewhat limiting.

UNDOCUMENTED SOUNDS

```
SOUND -1,2,0,0     a short belch
SOUND -1,13,0,0    defies description
```

MODIFICATIONS TO THE GAME "ANAGRAMS" TO GIVE SPEECH ECHOING OF INPUT AND OUTPUT

Insert the following lines into the program as listed in this issue:

```
385 FOR len=1 TO length:
    A$=MID$(shuffled$,len,1):
    SOUND -1,ASC(A$),0,0:
    SOUND -1,127,0,0:NEXT
405 SOUND -1,ASC(guess$(guess)),0,0
```

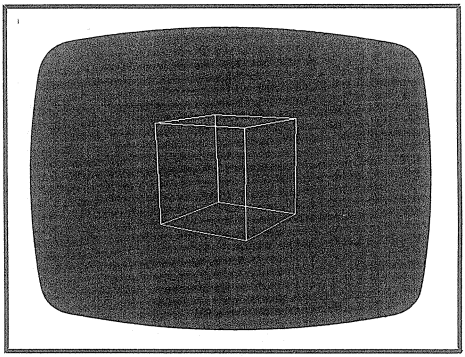
THIS PROGRAM LETS THE SPEECH CHIP SING (?) THE PRAISES OF THE BEEB

```
10 REPEAT
20 READ A
30 IF A>0 THEN SOUND -1,A,0,0
40 UNTIL A<1
50 DATA 267,127,160,209,
       159,202,179,0
```

THIS MONTH'S MAGAZINE CASSETTE CONTAINS AN AUDIO COPY OF THE SPOKEN VOCABULARY OF PHROM A.

3D ROTATION UPDATE

by Robert Alexander



BEEBUG vol 1 no 10 contained an interesting 3D rotation program which could be used to expand, contract or rotate any object. Here we show how to include the third basic graphic transformation, that of translation. The keys used to move the shape are :-

A - right C - up
B - left D - down

The new program lines listed below should be typed in after loading the original program.

```
1085 moveamnt%=50
1820 UNTIL key=13 OR key=93 OR key=127
 OR (key>134 AND key<140) OR (key>64 AN
D key<69)
1945 IF (key>64) AND (key<69) THEN PRO
Cmove
2400 DEF PROCmove
2410 REM Move object about viewing scr
een
2420 LOCAL movex%,movey%
2430 IF key=65 THEN movex%=moveamnt% E
LSE IF key=66 THEN movex%=-moveamnt% EL
SE IF key=67 THEN movey%=moveamnt% ELSE
 movey%=-moveamnt%
2440 FOR count%=1 TO points%
2450 X=X(count%):Y=Y(count%)
2460 X(count%)=X+movex%
2470 Y(count%)=Y+movey%
2480 NEXT count%
2490 ENDPROC
```

Program tested on O.S. 0·1 and 1·2

BAD PROGRAM LISTER (16k)

by Elizabeth Hanson

TESTED ON BASIC I
TESTED ON BASIC II

This program will list any Basic program (or any text file) stored in the computer, in a completely readable format, very similar to that produced by the LIST command. Its particular value is in listing programs which the command "LIST" will not work upon; ie programs which have become "damaged" in the computer, resulting in the familiar response, "Bad Program". It will even list the remains of a program that has been partly overwritten, by loading in another shorter program on top of it.

LISTER was written for use with the BEEBUG "RESCUE" program (Vol. 1, No. 8, p. 3), to look at and fix "bad programs". RESCUE is an extremely powerful utility which will heal "bad programs" miraculously. A copy of RESCUE is included on this month's magazine cassette.

LISTER allows you to examine a "bad program" before attempting to repair it, though it has many other applications, such as reading partially overwritten programs. LISTER works by looking at the Basic tokens as they are stored in memory, and converting them back to their original keyword format. The program is very small, just over one block, as it uses the look-up table in the Basic ROM at &806D (&8071 in Basic II) to make the conversions.

LISTER will not convert the line numbers following a GOTO, as these are stored in code within the computer to save space. Note that for the sake of speed (and the irregularity of the way the tokens are set up in memory), the word "AND" is printed as "ND". Also LISTER can be fooled by embedded Teletext colour commands, which it mistakes for keyword tokens, but this is not important for the uses for which LISTER is designed.

USING LISTER.

Type in the program and save a copy, before use, as you would any Basic program. If your machine has Basic II you will need to alter line 140 to:

```
140 DEFPROCL: K%=-1: T%=&8071
```

This is because the look-up table in Basic II starts at &8071 not &806D. To tell if you have Basic II, type REPORT followed by Return - a copyright message will appear; if the date is 1982 you have Basic II.

Assuming that you want to use LISTER to list a program already in the computer, you will need to change PAGE before you load in LISTER, this avoids overwriting the program already resident. Proceed as follows:

```
MODE7      <return>
PAGE=HIMEM-&200      <return>
CHAIN"LISTER"      <return>
```

LISTER will then run and print "Address ?" on the screen as a prompt for you to provide a starting address for it to list from. This will normally be &E00, or &1900 if you have a disc system, but you can type in any address that you want. The listing will then start and will print in 1k blocks, after each block it will beep, and wait for you to press any key. It will then print the current address in red followed by the next 1k of program, and so on.

```
  10 REM LISTER BY ELIZABETH HANSON
  20 REM VERSION 1A
  30 MODE7
  40 INPUT"Address",A$
  50 @%=1:J%=255:A%=EVAL(A$)
  60 REPEAT:PRINT'CHR$129;~A%
  70 FORI%=0TOJ%:X%=A%?I%
  80 IFX%=&0D PRINT'A%?(I%+1)*256+A%?
(I%+2);" ";:I%=I%+3:GOTO110
  90 IFX%>&7F ANDX%<&FF PROCL:GOTO110
 100 IFX%>31PRINTCHR$(X%);ELSEPRINT~X
%;
 110 NEXT
 120 A%=A%+J%+1:VDU7:B$=GET$:UNTILFAL
SE
 130 END
 140 DEFPROCL:K%=-1:T%=&806D
 150 REPEAT:K%=K%+1:Y%=T%?K%:UNTILY%=
X%ORK%>&300:L%=K%
 160 REPEAT:K%=K%-1:Z%=T%?K%:UNTILZ%>
&7F ORK%<0
 170 IFT%?(K%+2)<40K%=K%+3ELSEK%=K%+2
 180 REPEAT:PRINTCHR$(T%?K%);:K%=K%+1
:UNTILK%>=L%:ENDPROC
```

THE NEW EPSON AND SEIKOSHA PRINTERS

by David Tall and Adrian Calcraft

EPSON RX8Ø and FX8Ø
PRICE RX8Ø £34Ø inc VAT
FX8Ø £48Ø inc VAT
Reviewed by David Tall

Having decided to purchase an Epson MX8Ø printer of my own after a long and happy experience with the machine at work, it came as something of a shock to learn that it had been withdrawn from sale! In its place are two new models, the RX8Ø and the FX8Ø. Both greatly update the earlier printers, operating at twice the speed (16Ø characters per second), with an extra set of italic characters in ROM as well as the standard format. Each has a new elite typestyle in addition to the traditional Epson print, and both styles have a variety of sizes: normal, condensed, expanded and expanded/condensed, applying both to standard print and italic. The symbols may also be emphasised by repeating each dot one place to the right and double printed by repeating one position down, giving a wide range of print sizes and degrees of boldness. Though some combinations are not possible, there are sixteen to choose from, together with another sixteen corresponding styles arising from printing superscripts and subscripts, making 32 for each alphabet, a total of 64 including italic. Both printers can operate in high resolution bit-printing mode with the RX8Ø having the same facilities as the MX8Ø and the FX8Ø having a few extras. As with the two different versions of the MX8Ø, the cheaper of the two new models handles only pin-feed paper whilst the other also has friction feed enabling it to take individual sheets.

The additional cost of the FX8Ø arises partly from the inclusion of optional proportional spacing, but is mainly due to the incorporation of an area of RAM used either as a 2k print buffer or for programming up to 256 user-defined characters. It was this latter facility that caught my eye and, after shopping around to get a more acceptable price, the FX8Ø arrived

A few examples of FX80 printing:

This is normal print
and this is elite & italic.
This is proportional emphasised *italic*
and this is condensed superscript.
This is enlarged,
emphasised
& doubled.

Some examples of user-defined chara
αβ∂δεφγθι϶κλμνοπχρστυψωζηξ
ℵ∞ℂΔεΦΓΘ/φΘΛ-ℕØΠℚℝΣTUΨΩΞ¢ℤ
↑↓←→↔≈≡⊂⊃⊗∓∩\∀⇔≅≃∃≺≻/
∴∵♭∮∝⊕√⊂⊃≠÷⇐□±∘⇒≤≥⊥

ɔportional spacing but is ma
an area of RAM used either as a 2K
to 256 user-defined characters.
ught my eye and, after shopping ar
the FX80 arrived looking ver
romising to be far more versatile.
p the machine I was pleasantly sup
which initialize the printer
e top of the machine instead of
down and unscrewing the bottom.
ed one inch above the next print p
ch wasted half a page of paper eve
more sophisticated method of pape
ın chewing every piece of paper ins
'e failed to master it.
nable character facility is a joy.
:C PC8023BE-C printer because of it
ill-formed Greek alphabet on th
ekend's playing around with the Ep

looking very much like its predecessor, but promising to be far more versatile.

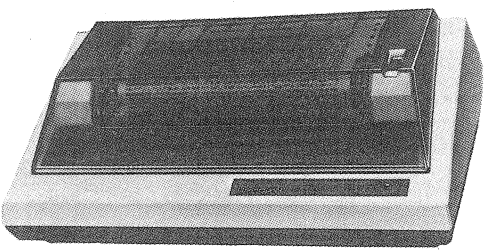
On setting up the machine I was pleasantly surprised to find that the dip-switches which initialize the printer were conveniently accessed from the top of the machine instead of the old system of turning it upside down and unscrewing the bottom. A new paper-tear bar is now situated one inch above the next print position instead of the old system which wasted half a page of paper every time. But this was countered by a more sophisticated method of paper insertion which seemed hell-bent on chewing every piece of paper inserted. Even after a week's use I have failed to master it.

The user-definable character facility is a joy. I had considered purchasing the NEC PC8023BE-C printer because of its Greek characters but, frankly, the ill-formed Greek alphabet on that printer is a disgrace. A weekend's playing around with the Epson user-definable capability gave me a satisfactory Greek alphabet and a whole array of mathematical symbols. Each character is defined on an array with 9 vertical positions and 11 horizontal ones. In practice only 8 vertical dots can be programmed, omitting the top or bottom line, and if the full 11 character width is utilized there will be no gaps between characters. When printed in the normal typeface the horizontal points are half-spaced, giving an acceptable quality, but it is not possible to program adjacent horizontal points to improve the definition. Even so, the facility is extremely versatile with programmable specifications for proportional spacing and all the printing options mentioned so far. The user definable RAM has enough space for two full keyboards of characters, making a total of four alternative alphabets available at any stage.

As far as I can see, every attempt has been made to make the control codes upward compatible from the old system so that existing software should run in conjunction with the new printer. The inability of the MX80 to maintain the correct page length in double printing has been rectified, allowing this bolder type-face to be used for word-processing. Of course, the print is still clearly dot-matrix quality and not up to the standard of daisy-wheels. Even so, in the few days I have used it I have become very impressed with its clean, no-nonsense elite typeface and its classier emphasised normal print. As a curiosity, the condensed superscript can be used with a narrow line-step to produce stupendously small print of amazing clarity.

The control codes sent to the printer to modify the typeface are logical but often cumbersome. To obtain a superscript requires the command ESC S Ø which is sent as VDU1,27,1,83,1,Ø and it must later be cancelled by ESC T (VDU1,27,1,82). So simple print modifications often require great strings of numbers to effect them. For word-processing a selection of these can be programmed on to the BBC user-defined keys but the present generation of word-processors for the BBC computer are not versatile enough to match the Epson printer, being unable to cope with proportional spacing or print of varying size. Clearly a later generation of software and printers will resolve these differences. Until that happens, in this price range the BBC computer and the Epson printer make a formidable combination.

SEIKOSHA GP-250X
PRICE £276 inc VAT
Reviewed by Adrian Calcraft



The 250X is the latest printer to be produced by Seikosha. It is compatible with the BBC micro, as are their previous two printers, the GP-80A and the GP-100A. The 100A was reviewed in the March issue of BEEBUG. The 250X is really a much improved version of the

100A, so to avoid repetition we will concentrate on the new features offered by the 250X.

The appearance of the 250X is identical to the 100A except that a small panel on the front containing two push buttons. The paper feed is of the "tractor" type, requiring special paper with holes. This is easily and cheaply available but does prevent single A4 sheets from being used in the normal way. I experienced no misfeeds while using the machine.

The printing method is impact dot matrix, which produces a noisy whine, which is reduced somewhat by the baffle. The characters are produced on a 5x8 matrix, allowing one substantial improvement over the 100A, "descenders". This means that lower case letters such as "p" "q" and "g" have tails which descend below the line. Although the difference is only one dot, it eases reading greatly and gives a more professional finish.

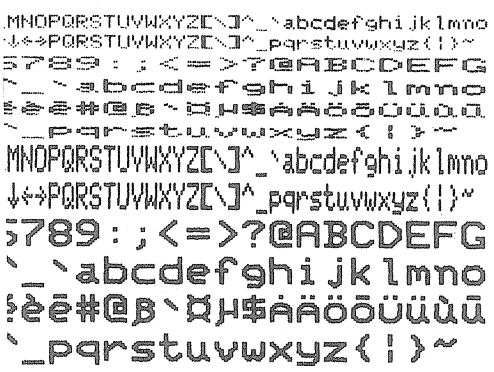
A set of 116 characters is available and the machine also has a switch to enable the choice of one of four different language character sets. In addition to this, a new feature is the ability to have up to 64 user-defined characters. Once defined these will remain available for use until the printer is switched off. Double height, double width or double height AND double width character print out is now possible, which is very handy for headings. The character underline feature available on other printers such as the Epson is unfortunately still not implemented on the new Seikosha.

Graphics printing is possible but requires a special screen dump program to control the printer. See this issue of BEEBUG for details of a 250X screen dump. Another improved feature is the print speed, the 100A printed at 30 characters per second, the 250X prints at 50. This speed difference is very noticeable, and is a welcome improvement. Unlike the 100A this printer can be used with either the serial interface or the parallel Centronics interface.

Other features which may occasionally be of use, include the ability to specify:

1) The actual space moved when a linefeed is issued.
2) The number of lines per page.
3) The amount of space between character cells.
4) A given print sequence which will be memorized and repeated at will.
5) The print position for character data.

As mentioned earlier the front face has a panel with two push buttons. These are marked STOP and RESET. The stop button will sound a warning buzzer and pause any further printing until the reset is pressed. This is handy to check the printer, if a misfeed or



The program works in any
is called at the same address in e
program exits,if it is called from

In MODEs 4 and 5,the p
dots) printout. In MODEs 0,1,2 it
printout. This means that the c
bottom of the screen, equivalent t
Y-coordinates from 0 to 63. Th
altering the variable topline% in
the top line (in text terms) that
1 (before assembly), then one line
If set to 2, then the top two line

If you are using a colou
odd colour changes before printi
restored as soon as the printer ha
let you choose one background c
else is printed, this dump prints
as greys by producing fine st
true/reverse video facility. Most

other problem is suspected. The buzzer will also sound and printing cease if the machine detects that it is about to run out of paper. Very useful as you can confidently leave the machine to print unattended. Printing can be resumed when paper has been inserted.

CONCLUSION

This printer is only slightly more expensive than the GP-100A and offers a lot of features which you will undoubtedly find useful from time to time. At the price of about £280 it is still one of the cheapest machines available, offering features usually found only on more expensive printers. The print quality is sufficient for home use (though still markedly inferior to the Epson). As a programmers tool this printer is very good indeed. I would have thought, however, that the print quality is not sufficiently good to be used outside the home or very small business environment.

One final point is that the manual supplied with the machine is very hard to follow, definitely user unfriendly and will probably prevent many people actually using several of the features they have paid for.

Thanks to Technomatic of Edgware Road, London for supplying our review machine, and to Clive Rodmell for his useful comments on the 250X and the screen dump.

BRAIN TEASER

by Gareth Suggett

Unpredictable Iterations

In previous issues we have asked you to investigate "iterative" processes ("Persistence" Vol 1, No 10, P8; "The 3x+1 Problem" Vol 2, No 2, P15) in which a number was chosen, and a rule applied repeatedly to it to generate a series of numbers.

Such a series can only exhibit three types of behaviour:

1. Reaching a single number, at which it 'sticks'.
2. Reaching a cycle of two or more numbers round which it goes forever.
3. Going on forever, never repeating itself.

(The third option is not possible in "Persistence", where the numbers available are finite in number - the n-digit numbers). The interest of such problems lies in the difficulty of predicting from the initial number, which of the three types of behaviour the sequence will exhibit.

Another such iteration is to start with any number. Stop if it is a palindrome (the same forwards or backwards), otherwise add to it the number formed by reversing its digits. Most numbers reach a palindrome after 2 or 3 steps, 89 and 177 give quite interesting results, but the problem is chiefly remarkable for the behaviour of 196.

Can you find other such unpredictable iterations? The field of iteration need not be numbers. Conway's "Game of Life" (Vol 1, No 10, P35), is a classic example. The two-dimensional nature of "Life" produces an added richness, but the three basic types of behaviour (stabilisation, oscillation or infinite development) remain.

Can anybody come up with a three (or four?) dimensional variation on this theme?

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MORE TELETEXT CODES FROM THE KEYBOARD (New O.S.) - M. Mc.Namara

When *FX4,2 is used to program the cursor and COPY keys, these keys behave like the function keys do when used with shift or control, thus shift-COPY generates code 139 - start box and Ctrl-left-cursor generates black background.

Program tested on O.S. 0·1 and 1·2

USING FILES (PART 4)

by Sheridan Williams

We continue our series on using files by giving two demonstration programs that use random access files. The second of these could form the basis of a very sophisticated business system, possibly allowing your BBC micro to earn its keep if you have bought it for business purposes.

One way to envisage a random access file, is to think of it as a large array that doesn't use any of the Beeb's precious memory (RAM). Like an array you can access any element of the file, however the access time is longer than it would be with an array, taking anything up to 1 second in the worst case. This is a small price to pay for an array that could be perhaps 40,000 elements long (using 'integers' on a 200k per side disc).

THE PTR FUNCTION

The PTR function is the key to random access files. By using PTR followed by PRINT# or INPUT# we can read or write to ANYWHERE within the file, instead of the next immediate record as we had to with serial files. Care must be exercised, because a miscalculation by only ONE byte will cause an error and hence make an item of data unreadable. In the example below we are going to write 1000 integers to the file. If you read page 330 of the User Guide you will see that integers occupy 5 bytes on the file. Armed with that knowledge we set about writing our numbers 5 bytes apart: for example if we write the first number at position 1 on the file, we must write the second number at position 6, and so on. In fact the simple formula i%*5 calculates the start address of each number: this gives addresses 1, 5, 10 etc for records 1, 2, and 3 etc.

DEMONSTRATION PROGRAM ONE

So here is the program that first writes 1000 random numbers to a disc file, it then allows you to access any one of the 1000 random numbers immediately. You can witness the disc buffer in action here: try accessing record 100, then record 110, and you will hear no disc movement. However, if you access record 1 followed by record 100 you will hear the disc start. As the disc buffer is 256 bytes you will find no extra disc reads are necessary if you read record 1 followed by record 50 (because 50 records at 5 bytes each is only 250 bytes).

```
  10  CLS
  20  c=OPENIN"RANDOM"
  30  IF c=0
c=OPENOUT"RANDOM":CLOSE#c:GOTO 20
  40  n%=0
  50  PRINT TAB(8,10)"Writing:"
  60  FOR i%=1 TO 1000
  70    n%=RND(10)+n%
  80    PTR#c=i%*5:PRINT#c,n%
  90    PRINT TAB(18,10);i%
 100  NEXT i%
 110  REPEAT
 120    INPUT"Which record do you want
to see ",i%
 130    IF i%<1 OR i%>1000 CLOSE#c:END
 140    PTR#c=i%*5:INPUT#c,n%
 150    PRINT"Record ";i%;" is:";n%
 160  UNTIL FALSE
```

Here is a brief description of what each statement does:-

20 attempts to open the file for INPUT. If a file is to be used for both reading and writing it must be opened for INPUT and not for OUTPUT. However, it is not possible to open a file for INPUT if it doesn't already exist. Whether the file exists or not is detected in line 30 because the channel allocated will be zero if the file doesn't already exist. In that case the file is opened for OUTPUT (thus creating it) and closed immediately. Now when we re-execute line 20 the file exists.

40 Set last random number generated to zero.

60 Start loop at one.

70 Generate a new random number (n%) that is greater than the last.

80 Move pointer to position i%*5 and PRINT the random number to the file.

90 Display on the screen the number of

times round the loop.
100 Increment the loop and continue.
140 Move pointer to the required position and INPUT the number from the file.

DEMONSTRATION PROGRAM TWO

The following program uses a random access file to hold the "People" file. (Defined in "Files Part 3"). Use this program as a basis for study as it has lots of potential. Each person is given a number between 1 and 100, (an empty 200k disc would hold nearly 6000 records, but restricting it to just 100 gives ample demonstration capabilities.)

The program has four operations - initialise, write, read and end.

Initialise - This will clear (erase) a record or set of records by writing spaces in place of data. It is essential to initialise records 1 to 100 before using any of the other options as this will check that there is enough room on the disc.

Write - Use this option to enter data about a person. You will be asked for the record number which is the position on the file that you want the record placed.

Read - Use this option to examine what is in any particular record. (You must not read a record that has not been initialised, otherwise you will get a read error.)

End - You MUST use this option to stop the program, otherwise the most recent contents of the disc buffer will not be stored on disc.

NOTES ON THE PROGRAM

Lines 30, 40 - As explained in the program above this is a precaution in case the file doesn't yet exist.

Line 50 - rl represents the record length. We set it to 34 in our case because this is the length of the data plus TWO characters (as explained in parts 2 and 3). We need rl in order to find the start address for each record. For example in line 2140 we want to initialise record 12 say, then we print on 32 spaces starting in position 12x34.

Lines 1000-1040 - This function prompts you to enter a record number, and returns with this value.

Line 6000 - This function is required to pad out each field to its specified length.

Line 7065 - This string makes the printout much clearer.

Line 8000 - Error 6 is generated if the file is not initialised when an INPUT# takes place.

Line 8010 - If the user tries to end by pressing 'Escape' he is reminded to finish via option 4.

```
   10  REM BEEBUG demonstration of
random access files
   20  ON ERROR GOTO 8000
   25  filename$="PEOPLE"
   30  c=OPENIN filename$
   40  IF c=0 c=OPENOUT
filename$:CLOSE#c:GOTO 30
   50  rl=34
   60  nul$="":REM no spaces between
quotes
   70  sp$=" ":REM one space between
quotes
  100  REPEAT:CLS
  110    PRINT TAB(7,5)"1. Initialise
the file"
  120    PRINT TAB(7,6)"2. Write a
record"
  130    PRINT TAB(7,7)"3. Read a
record"
  140    PRINT TAB(7,8)"4. End."
  150    PRINTTAB(4,10);CHR$131;"WHICH
OPTION? ";
  160    option=GET-48
  170    PRINT ;option
  180    IF option=1 PROCinitialise
  190    IF option=2 PROCwrite
  200    IF option=3 PROCread
  210    IF option=4 CLOSE#c:END
  220    UNTIL FALSE
  999  END
 1000  DEF FNrecord_number
 1010  REPEAT
 1020    INPUT"Record number (1-100)
",record%
 1030    UNTIL record%>=1 AND
record%<=100
 1040  =record%
```

```
 1999
 2000  DEF PROCinitialise
 2010  LOCAL r%,from%,to%
 2020  rec$=STRING$(32,sp$)
 2100  PRINT'"From
";:from%=FNrecord_number
 2110  PRINT"To ";:to%=FNrecord_number
 2120  CLS:PRINT TAB(7,10);"I N I T I A
L I S I N G"
 2130  FOR r%=from% TO to%
 2140    PTR#c=r%*rl:PRINT#c,rec$
 2150    PRINT TAB(19,12);r%
 2160    NEXT r%
 2170  ENDPROC
 2999
 3000  DEF PROCwrite
 3020  INPUT'"Surname",sur$
 3025  IF sur$=nul$ ENDPROC
 3028  sur$=FNpad(sur$,15)
 3030  INPUT"Title  ",title$
 3035  IF title$=nul$ THEN 3020
 3038  title$=FNpad(title$,10)
 3040  REPEAT
 3050    INPUT"Sex (M/F)",sex$
 3060    UNTIL sex$="M" OR sex$="F" OR
sex$=nul$
 3065  IF sex$=nul$ THEN 3030
 3080  INPUT"Date of birth
(DDMMYY)",dob$
 3090  IF dob$=nul$ THEN 3040
 3100  dob$=FNpad(dob$,6)
 3110  rec$=sur$+title$+sex$+dob$
 3120
PTR#c=FNrecord_number*rl:PRINT#c,rec$
 3200  ENDPROC
 3999
 6000  DEF
FNpad(Q$,L)=LEFT$(Q$+STRING$(L,sp$),L)
 6999
 7000  DEF PROCread:LOCAL q
 7010
PTR#c=FNrecord_number*rl:INPUT#c,rec$
 7030  sur$  =LEFT$(rec$,15)
 7040  title$=MID$(rec$,16,10)
 7050  sex$  =MID$(rec$,26,1)
 7060  dob$  =MID$(rec$,27,6)
 7065  L$="<"+CHR$13+CHR$10+">"
 7070  PRINT"Record
contains"'">"sur$L$title$L$sex$L$dob$"<
"
 7080  PRINT"Press any key when
ready":q=GET
 7090  ENDPROC
 8000  ON ERROR OFF
 8005  IF ERR=6 PRINT'"File not
initialised at this point":GOTO 8030
 8010  IF ERR<>17 REPORT:PRINT "at line
no. ";ERL:CLOSE#17:END
 8020  PRINT'"You must end using
option 4"
 8030  q=INKEY(200):GOTO 100
```

Using this program it is possible to record data on thousands of people permanently. Records can be updated, viewed or erased quite simply. The principles behind this program are fundamental to virtually all file processing.

NEXT MONTH

We shall look at methods of calculating the addresses of records on file. Also we will look at index-sequential files as an alternative to random access files.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

1001 USES FOR A SQUARE BRACKET - Malcolm Davis.

Did you know that typing '[' then <return> prints out the current value of the assembler's program counter in hex? Try it on power up and you should get '0000'. It can be altered by setting P% and is not affected by soft or hard resets. The longhand way to find its value in hex is to type PRINT ~P% <return>. Also CTRL [is the same as pressing the Escape key!

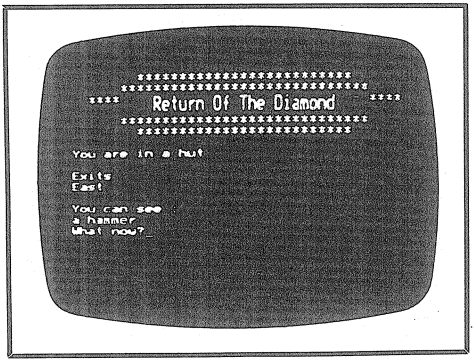
CIRCULAR CIRCLES ON A TELEVISION - Richard Russell

When writing graphics programs, particularly those using geometric shapes, bear in mind that the BBC Micro's "graphic units" are not square when displayed on a properly adjusted TV set or monitor. In fact they correspond to rectangles whose height/width ratio is approximately 1:1.084. As an example, if you want to draw an accurate circle the program must be written as if it is drawing an ellipse whose horizontal axis is 1.084 times its vertical axis when measured in graphics units.
[The above applies to a correctly adjusted television screen. Some monitors have different aspect ratios, and in any case it is sometimes possible to alter the height control of the TV/monitor to correct this. Ed]

EXPANDING RETURN OF THE DIAMOND

by R. McGregor and Alan Webster

An adventure is a game in which you travel about in a world described to you by the computer. The computer displays where you are, what you can see and in which directions you can move. You use commands such as "GET", "KILL" and "INVENTORY" and move using directions like "N", "E", "S", "W" and sometimes "U" for up and "D" for down.



The first thing you have to do when writing an adventure program is to make up a map and give each position a number. Figure 1 shows the map for the small adventure, "RETURN OF THE DIAMOND", published last month. As its name suggests your mission is to find the "great diamond", and return it to its rightful owner at the "diamond castle". The words in capitals in the figure are the place names, and those in lower case are the items which are originally found there. So as not to spoil the game completely however, we have not revealed where the diamond is located.

If you play an adventure game such as Return of the Diamond, you will almost certainly need to produce a map like this as you begin to find your way around. It is also the starting point when writing such a game. You must decide on the number of locations, the way in which they are connected, and the objects to be found in each. Probably the best way to illustrate how the program has been constructed is to show how to expand the existing adventure. This is an interesting task in its own right, and the way in which the program has been constructed allows relatively easy expansion to a considerable degree of complexity.

The details which follow will make more sense if you have taken a brief look at the program (this was listed last month, and also appears on last month's magazine cassette), and as an aid, a list of procedures is given here. These are called from the main program loop on lines 10 to 140.

EXPANDING THE PROGRAM.

Let us see how to incorporate the following modifications:

1. Add an extra location
2. Add an extra item
3. Add an extra command

ADDING A LOCATION

To add an extra location to the game we need to know where we are going to put it, give it a number and define the directions in which it can be accessed. Suppose the new location (location 10) is to the right of the forest (location 6), and connects to the forest only.

First we have to increase the array size for the number of locations. This is at line 180. If we are adding just the one location to our original program which had nine locations then we need to change lines 180 and 190 to:

```
180 DIM place$(10)
190 FOR i=1 TO 10
```

Secondly we need to create a description for our new location. This I shall call the Library. The following change is needed to line 260:

```
260 DATA in a dark passage,in the
    Library
```

Thirdly we need to alter the direction pointers to take account of this new location. To do this replace the value 9 with 10 in lines 270 and 280. Thus:

```
270 DIM newpos(10,4)
280 FOR i=1 TO 10
```

It is also necessary to alter the data at line 340 so that the EAST pointer for location 6 points to location 10. The pointers for locations 4, 5 and 6 are all held in line 340. The first

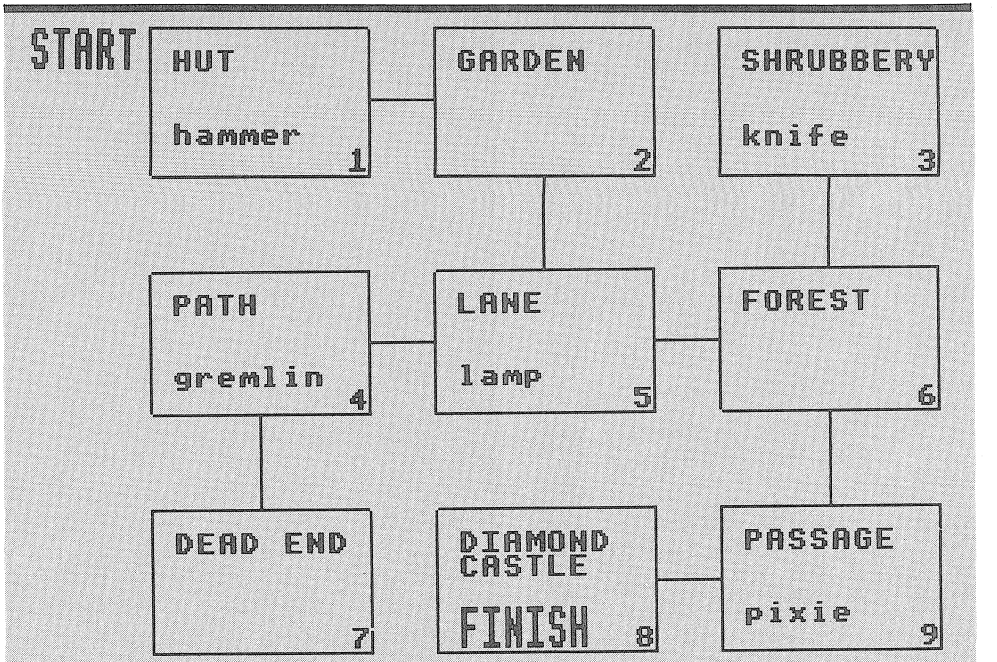


Fig 1 MAP OF RETURN OF THE DIAMOND

four numbers are the pointers for location 4 (N, E, S and W). The first is zero. This means that you cannot go North from location 4. The second is 5, meaning that you can go East from location 4, and will arrive at location 5, and so on. The middle four pointers on line 340 are for location five, and the last four are for location six. It is this that we must alter to indicate that location 6 leads East to the new location 10. Doing this gives us:

```
340 DATA 0,5,7,0,2,6,0,4,3,10,9,5
```

Now we need to create the pointers for location 10. This we shall do on line 355 by typing:

```
355 DATA 0,0,0,6
```

The first 3 zeros indicate that there are no exits to the North, East or South. Item 4 indicates that going West from location 10 takes you to location 6.

ADDING AN ITEM

The second part of our expansion is to introduce a new item. This item is to be a BEEBUG magazine and we will put it in the library...where else! To add this extra item we need to expand the arrays which hold the relevant information. These arrays are at line 360 and hold the detailed description, the brief description and the location number respectively.

```
360 DIM item$(7),itemname$(7),
    itempos(7)
```

and line 370 needs to be changed to:

```
370 FOR i=1 TO 7
```

Next we need to give the extra data which was outlined above. To add this we shall insert it at line 455.

```
455 DATA a precious BEEBUG magazine,
BEEBUG,10
```

and alter these further lines:

```
930 FOR i=1 TO 7
1330 FOR i=1 TO 7
1660 UNTIL no>0 OR i=7
```

To give the BEEBUG magazine the importance it deserves, we can make a condition that you can only finish the game if you have it in your possession. To do this we need to change the winning condition at line 1170 to:

```
1170 won=((position=8 AND itempos
     (2)=8) ANDitempos(7)=0)
```

and because of this we need to prolong our lamp life, so change line 610 to:

```
610 reallit=3.4
```

ADDING A COMMAND

The third part of the expansion is to add an extra command. The command that I shall add is one which enables the player to quit the game at any stage, whereupon the computer tells the player his score and then ends. To do this we need to expand the array which holds the commands. Alter lines 460 and 470 to:

```
460 DIM com$(8)
470 FOR i=1 TO 8
```

Now we need to add our command to the data list in line 500.

```
500 DATA GET,TAKE,ON,LIGHT,
    OFF,DROP,KILL,QUIT
```

and add a line 1055 to deal with the command when it is met.

```
1055 IF c$="QUIT" PRINT:PRINT score:
     PRINT "Goodbye":END
```

That completes our simple expansion, but with over 16k of space left on a 32k machine, you have plenty of room for your own expansion. You could for example make the castle multi-roomed, and introduce new problems to solve and dangers to overcome. Alternatively a look at the way in which the adventure is written may give you ideas for writing your own.

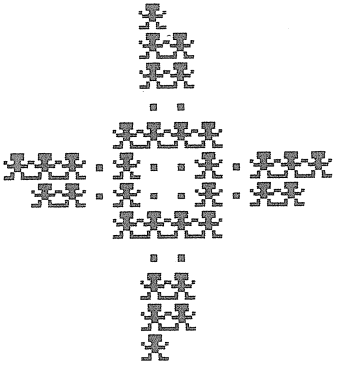
ADVENTURE PROCEDURES

Set-up	Sets up the arrays and initialises the variables
title	Prints the title
look	Prints the current position, with room description, exits and items
analyse	Tests for a move, look, inventory, score, quit, or move command
other-commands	This deals with the rest of the commands
time-passing	Decreases score, increases moves and deals with the lamp
move	Moves the player
inventory	Tells the player of all the items that he is carrying
take	Takes selected item
light	Lights the lamp
off	Turns the lamp off
drop	Drops the selected item
kill	Tests if the player has met a gremlin or a pixie
kill-gremlin	Kills the gremlin
kill-pixie	Kills the pixie
finish	End of game

LIFE COMPETITION WINNERS

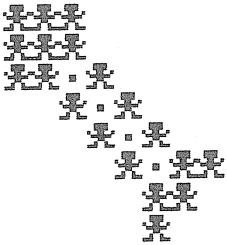
In Vol. 1 No. 10 we included a 'LIFE' program, to simulate the growth of a colony of frogs. We set a competition for the most interesting pattern within 100 generations. As was made clear, most interesting means most interesting to the editor. We have now sorted through the entry and have selected the winners of the prizes.

Richard Porter - Prize £20



Place this formation centrally.

Steve Broderick - Prize £10



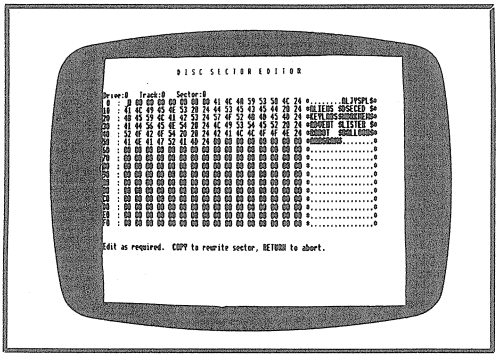
Place near the bottom right of the screen.

O.S.1.2 ONLY

DISC SECTOR EDITOR (32k)

Program by Mike Ball

This excellent utility allows you to read, edit and write to and from any part of a disc directly. You can as a consequence recover files lost by accidental deleting, by partial overwriting, or by certain kinds of disc or drive fault.



When you run the program it asks you first which drive you are using, and then which track and sector of the disc you wish to look at. Discs have 10 sectors per track, and either 40 or 80 tracks. There are thus either 400 or 800 sectors on each side of a disc. The syntax is Track, Sector. If you wish to look at sector 3 of track 2, enter "2,3", and so on. Alternatively you can Tab along and enter a filename in place of a track and sector. On pressing Return, you will see the 256 bytes of the named sector (or the first sector of the named file) displayed on the screen in an 16 x 16 format in hex, and alongside in ASCII. You can then use the cursor keys to move to any chosen byte (either in the hex or the ASCII table), and either may be edited as normal. When you do this you will see the corresponding change in the other table. When you have finished editing a sector you can choose to abort, or to write back the edited sector to the disc, so as to replace the old sector with the new.

CATALOGUE SECTORS 0 AND 1

Perhaps the most interesting two sectors of the disc are 0, and 1 of track 0. These contain the catalogue information, including the names of all files on the disc and pointers to the sectors containing those files. It is these which need to be edited to retrieve lost programs or data (assuming of course that all or part of the contents of the file still exist).

Table 1 gives the format of sector 0. Its 256 bytes are completely filled by the 31 possible file names of 8 bytes each plus the first 8 bytes of the disc title. In our view it would have been useful to take up two sectors with this information so as to allow more than 31 files per disc, and file names of more than 7 characters (the eighth is the directory).

As an experiment with the editor, run the program, then insert a disc with a number of files on it, and call up sector 0 of track 0. You will see the various file names listed in ASCII. You may edit these as desired. For example you could generate a red title (in mode 7) for your disc by making the first byte on the disc &81. If you press the Copy key, the disc will be written to with the editings you have created, and if you exit the program and type
*CAT <Return> you should see a red disc title. File names should not be coloured because this confuses the disc filing system. If you wish to make your disc uncataloguable, you can do this by replacing the first 4 bytes of sector zero with &17, 0, 0, 0. The effect of this is to turn off the VDU controller chip at any time when *CAT is performed. Pressing Break will recover the chip, but without allowing the disc to be read.

RECOVERING LOST FILES

Files can become lost or corrupted for a number of reasons - they may have been accidentally erased, or there may be a disc fault. In either case the disc editor can enable you to retrieve some or all of the file. The reason for this is that deleting a file simply clears sectors 0 and 1 of track 0 of

all information about it, leaving the file intact somewhere on the disc. Of course if you save further data, it may well be overwritten. The first thing that you need to know when attempting to recover a lost file is at what sector the file begins, and how long it is. Both these pieces of information can be ascertained by typing *INFO filename - before the file is corrupted; and it may well be worth printing out *INFO *.*<Return> on important discs whenever new files are added. If you do not have this information, then you must search the disc, sector by sector until you find the start sector of the lost file, and its approximate length. You then need to create a new filename and appropriate pointers in sectors Ø & 1 of track zero using the editor. Sector Ø is easy, and may be accomplished by referring to table 1, and examining a disc with known filenames. Sector 1 is more complex. Table 2 gives the format. You will notice that the number of sectors on the disc may be a 1Ø bit number, and that the top two bits are stored in a different byte. To create a new file, you must increase byte Ø5 by 8 to correct the file count. Next you will need to add 8 bytes of data to the appropriate storage map. Table 2 gives the location of these 8 bytes, while table 3 gives their contents.

In order to help clarify this, we give below a step-by-step instruction list for recovering a lost file once you know its start sector (a file always starts at the beginning of a sector), and its approximate length in bytes.

1) Find the first free 8-byte filename block in sector Ø and edit in the name and extension that you require.
2) Re-write sector Ø and then bring in sector 1. Increase the count of the number of files (byte &Ø5) by 8.
3) Find the first free 8-byte storage map, which should have the same initial byte address as the filename block used in sector Ø.
4) Enter the appropriate Load address in the first two bytes of this block. This will be &19ØØ for Basic and &ØØØØ for data files.
5) Enter the Execution address in the third and fourth bytes. Again this would be zero for a data file but &8Ø1F for a Basic file.
6) Enter the Length of the file (in bytes) in the fifth and sixth bytes. Note that until a 16-bit second processor becomes available this value is not going to be greater than 64k (ie. &FFFF).
7) Now the only concern is with the least two significant bits of the seventh byte and the whole of the eighth byte, which hold a 1Ø-bit value corresponding to the start sector for the file. If the file started on sector 5 of track 3Ø the start sector address would be 3Ø*1Ø+5=3Ø5=&131. The binary equivalent of this is Ø1 ØØ11 ØØØ1 so it is necessary to make sure that the bottom two bits of the seventh byte hold the binary value Ø1 (which will normally result in the byte &Ø1 existing in this location as the rest of the byte is usually zero), and that the value &31 (i.e. ØØ11 ØØØ1) is placed into the eighth location.

Re-writing sector 1, followed by a catalogue of the disc should now see your program or data returned. In a forthcoming issue we will be looking at some utilities to make this process more of an automated one, and to make the search for the beginning of your file easier.

TABLE 1

Sector Ø:

Byte	Use
ØØ-Ø7	: First 8 characters of disc title (padded with spaces).
Ø8-ØE	: Name of first file (padded with spaces)
ØF	: and its directory letter.
1Ø-16	: Name of second file
17	: and its directory letter.
18-1E	: Name of third file .....
1F	(and so on up to 31 files)

TABLE 2

Sector 1:

Byte	Use
00-03	: Last 4 characters of disc title
04	: Count of no. of write operations made to the disc
05	: 8 * no. of catalogue entries. A file name which exists in sector 0 will not be recognised until this value is set correctly.
06	: Individual bits have significance: 7 - 6 - 5,4 - start up option (!BOOT) 3 - 2 - 1,0 - No. of sectors on disc (2 MSBits of 10-bit number)
07	: No. of sectors on disc (8 LSBits of 10-bit number)
08-0F	: Storage map (see later) for first file.
10-17	: Storage map for second file.
18-1F	: Storage map for third file ..... (and so on up to 31 files)

TABLE 3

Byte	Use
08-09	: Load address. LSB first. (16 LSBits of an 18 bit address). This address would normally be zero for a data file and &1900 for a BASIC program.
0A-0B	: Execution address. LSB first. (16 LSBits of an 18 bit address). This address would normally be zero for a data file and &801F for a BASIC program.
0C-0D	: Length of file in bytes. LSB first. (16 LSBits of an 18 bit value).
0E	: Individual bits have significance: 7,6 - Execution address. 2 MSBits of an 18 bit address). For a single processor system this would be zero. 5,4 - Length of file in bytes. 2 MSBits of an 18 bit value) 3,2 - Load address. 2 MSBits of an 18 bit address). For a single processor system this would be zero. 1,0 - Start sector for file (2 MSBits of 10-bit number)
0F	: Start sector (8 LSBits of 10-bit number)

If the first file were a BASIC program starting on track 0, sector 2, with a length of 32 bytes, the above byte addresses would contain the following:

byte - 08 09 0A 0B 0C 0D 0E 0F
value - 00 19 1F 80 20 00 00 02

```
  10 REM Disc Sector Editor
  20 REM M.Ball VERSION 1A
  30
  40 LOMEM=LOMEM+8-(LOMEM MOD 8)
  50 DIM blc% 256,dir% 256,blk% 12
  60 D%=0:rcd%=0
  70 rwr%=FALSE
  80 hex$="0123456789ABCDEF"
  90 TRK%=0:SCT%=0:RSCT%=0
 100 fnm$=STRING$(10," ")
 110 A$=STRING$(16," ")
 120
 130 MODE 5
 140 PROCintro
 150 MODE 0
 160 VDU19,0,6;0;19,1,0;0;
 170 VDU23,0,10,0,0;0;0;
 180
 190 PRINTTAB(20,2)"D I S C  S E C T O
 R  E D I T O R"
 200 *FX4,1
 210 *FX12,4
 220
 230 REPEAT
 240 ON ERROR GOTO 400
 250 PROCsget
 260 PROCread(TRK%,SCT%,blc%)
 270 IF rcd%<>0 THEN SOUND1,-15,10,20:
PRINTTAB(40,5);"Disc error ";rcd%:GOTO3
60 ELSE PRINTTAB(40,5); STRING$(20," ")
```

```
  280 PROCdump
  290 PROCedit
  300 PROCwd
  310 IF rwr% THEN PROCwrite(TRK%,SCT%,
blc%)
  320 VDU28,0,31,79,0
  330 IF rwr% AND rcd%<>0 THEN SOUND1,-
15,10,20:PRINTTAB(40,5);"Disc error ";r
cd%:GOTO280 ELSE PRINTTAB(40,5); STRING
$(20," ")
  340 PROCwd
  350 IF rwr% THEN PRINT"S e c t o r  r
 e w r i t t e n":SOUND1,-15,150,5:A=IN
KEY(300)
  360 UNTIL FALSE
  370 END
  380
  390 REM End of Program
  400 VDU6:*FX4,0
  410 *FX12,0
  420 *FX229,0
  430 MODE7
  440 REPORT:PRINT" at ";ERL
  450 END
  460
  470 DEF PROCwd
  480 VDU 28,0,31,79,24:CLS
  490 ENDPROC
  500
  510 DEF PROCread(trk%,sect%,blc%)
  520 blk%?6=&53
  530 PROCioset
  540 rcd%=blk%?10
  550 ENDPROC
  560
  570 DEF PROCwrite(trk%,sect%,blc%)
  580 blk%?6=&4B
  590 PROCioset
  600 rcd%=blk%?10
  610 ENDPROC
  620
  630 DEF PROCioset
  640 rcd%=0 : ?blk%=D% : blk%!1=blc%
  650 blk%?5=3 : blk%?7=trk%
  660 blk%?8=sect% : blk%?9=&21
  670 X%=blk%MOD256 : Y%=blk%DIV256
  680 A%=&7F : CALL &FFF1
  690 ENDPROC
  700
  710 DEF PROCdump
  720 VDU 28,0,22,79,6:CLS
  730 E%=blc%
  740 F%=blc%+256
  750 @%=2
  760 REPEAT
  770 PRINT~E%-blc%;:PRINT "  : ";
  780 A$=""
  790 FOR X%=E% TO E%+15
  800 Y%=?X%
  810 IF (Y%<32)OR(Y%>122) THEN A$=A$+"
." ELSE A$=A$+CHR$(Y%)
  820 IF Y%>15 THEN PRINT;~Y%;" "; ELSE
PRINT;"0";~Y%;" ";
  830 NEXT
  840 PRINT "*"A$"*"
  850 E%=E%+16
  860 UNTIL E%=F%
  870 ENDPROC
  880
  890 DEFPROCsget
  900 RSCT%=TRK%*10+SCT%
  910 PROCwd
  920 PRINT"Hit 'N' for next sequential
 sector, or"
  930 PRINT"TAB to required input field
, overtype and hit return."
  940 PRINT'"TRACK/SECTOR (Dec)";SPC(4)
;"RELATIVE SECTOR (Hex)";SPC(11);"FILEN
AME (dir.name)"
  950 PRINTTAB(4);TRK%;",";SCT%;TAB(29)
;~RSCT%;TAB(54);
  960 IF LEFT$(fnm$,1)="*"THEN PRINT fn
m$ ELSE PRINT STRING$(20," ")
  970 VDU 11:PRINT TAB(4);:C%=4:fnm$=""
  980 A=GET:IF A=9 THEN PRINT TAB((POS+
25)MOD75,4);:C%=POS:GOTO980
  990 IF A=13 OR A=27 OR A=127 OR (A>13
4 AND A<140) THEN 980
 1000 IF A=78 OR A=110 THEN RSCT%=RSCT%
+1:TRK%=RSCT%DIV10:SCT%=RSCT%MOD10:GOTO
1060
 1010 A$=CHR$(A):PRINT A$;SPC(10);STRIN
G$(10,CHR$(8));
 1020 INPUT LINE "" ASK$:A$=A$+ASK$
 1030 ok=FALSE
 1040 IF C%=54 THEN PROCipfn ELSE IF C%
=29 THEN PROCiprl ELSE PROCipts
 1050 IF NOT ok THEN VDU7:GOTO910
 1060 VDU26:PRINTTAB(0,5)"Drive:";D%;SP
C(3)"Track:";TRK%;SPC(3);"Sector:";SCT%
;SPC(4);
 1070 ENDPROC
 1080
 1090 DEFPROCipfn
 1100 fnm$=A$
 1110 IFLEN(A$)<3 OR MID$(A$,2,1)<>"."
THEN fnm$="*INVALID*":ENDPROC
 1120 PROCread(0,0,dir%)
 1130 DIR$=LEFT$(A$,1):F$=MID$(A$,3,7):
IFLEN(F$)<>7 THEN F$=F$+LEFT$(STRING$(6
," "),7-LEN(F$))
 1140 off=0
 1150 FOR I%=dir%+8 TO dir%+248 STEP 8
 1160 svd=I%?7 : I%?7=&0D : fnt$=$I% :
I%?7=svd
 1170 IF (F$=fnt$) AND (DIR$=CHR$(svd))
 THEN off=I%-dir% : I%=99999
 1180 NEXT I%
 1190 IF off=0 THEN fnm$="*NOT FOUND*":
ENDPROC
 1200 PROCread(0,1,dir%)
 1210 I=dir%+off+6
```

```
 1220 RSCT%=(?I AND &01)*256 + ?(I+1)
 1230 TRK%=RSCT% DIV 10
 1240 SCT%=RSCT% MOD 10
 1250 ok=TRUE
 1260 ENDPROC
 1270
 1280 DEFPROCiprl
 1290 RSCT%=EVAL("&"+A$)
 1300 TRK%=RSCT% DIV 10
 1310 SCT%=RSCT% MOD 10
 1320 ok=TRUE
 1330 ENDPROC
 1340
 1350 DEFPROCipts
 1360 A$=A$+CHR$13
 1370 FORI%=1TOLEN(A$):A%=138:X%=0:Y%=A
SC(MID$(A$,I%,1)):CALL&FFF4:NEXT
 1380 PRINTTAB(4,4);:INPUT""TRK%,SCT%
 1390 ok=TRUE
 1400 ENDPROC
 1410
 1420 DEFPROCedit
 1430 PROCwd
 1440 PRINT"Edit as required.  COPY to
rewrite sector, RETURN to abort.";
 1450 VDU 28,6,22,70,6,30
 1460 *FX229,1
 1470 REPEAT
 1480 K%=GET
 1490 IF K%=13 THEN rwr%=FALSE:GOTO1600
 1500 IF K%=135 THEN rwr%=TRUE:GOTO1600
 1510 IF K%<32 THEN GOTO1600
 1520 IF K%=136 THEN IF (VPOS>0)OR((VPO
S=0)AND(POS>0)) THEN VDU8:GOTO1600
 1530 IF K%=137 THEN IF (VPOS<15)OR((VP
OS=15)AND(POS<64)) THEN VDU9:GOTO1600
 1540 IF K%=138 THEN IF VPOS<15 THEN VD
U10:GOTO1600
 1550 IF K%=139 THEN IF VPOS<>0 THEN VD
U11:GOTO1600
 1560 REM Not a control char, so this
 1570 REM must be attempt to alter data.
 1580 IF (POS<47)AND(FNs_ch<>" ") THEN
PROCxhex
 1590 IF (POS>48)AND(POS<65) THEN PROCxchr
 1600 UNTIL (K%=13)OR(K%=135)
 1610 IF K%=13 THEN CLS:PRINT TAB(15,8)
"E D I T I N G   A B O R T E D"
 1620 *FX229,0
 1630 ENDPROC
 1640
 1650 DEFFNs_ch
 1660 A%=135
 1670 C%=(USR(&FFF4) AND &FFFF) DIV &100
 1680 =CHR$(C%)
 1690
 1700 DEFPROCxhex
 1710 IF INSTR(hex$,CHR$(K%))=0 THEN SO
UND 1,-15,100,8:ENDPROC
 1720 cux=POS:cuy=VPOS
 1730 REM Now alter screen
 1740 VDU K%,8
 1750 IF (POS MOD 3)>0 THEN VDU 8 :REM
Move to left digit
 1760 ldig$=FNs_ch
 1770 VDU 9
 1780 rdig$=FNs_ch
 1790 REM Get new value for byte
 1800 new%=(INSTR(hex$,ldig$)-1)*16 + I
NSTR(hex$,rdig$)-1
 1810 REM Get offset within sector
 1820 ofs%=VPOS*16 + (POS DIV 3)
 1830 blc%?ofs%=new%
 1840 REM Get posn of char on screen
 1850 cpos%=49+(POS DIV 3)
 1860 VDU 31,cpos%,VPOS :REM Posn for p
rint
 1870 IF (new%<32)OR(new%>122) THEN PRI
NT".";: ELSE PRINTCHR$(new%);
 1880 VDU 31,cux,cuy
 1890 VDU 9
 1900 ENDPROC
 1910
 1920 DEFPROCxchr
 1930 cux=POS:cuy=VPOS
 1940 REM Get offset within sector
 1950 ofs%=VPOS*16 + POS-49
 1960 blc%?ofs%=K%
 1970 REM Get posn to print hex
 1980 cpos%=(POS-49)*3
 1990 VDU 31,cpos%,VPOS
 2000 IF K%>15 THEN PRINT;~K%;" "; ELSE
 PRINT;"0";~K%;" ";
 2010 VDU 31,cux,cuy
 2020 IF (K%<32)OR(K%>122) THEN PRINT".
";: ELSE PRINTCHR$(K%);
 2030 ENDPROC
 2040
 2050 DEFPROCintro
 2060 COLOUR129:CLS
 2070 VDU 28,2,29,17,2
 2080 COLOUR130:CLS
 2090 VDU 28,4,27,15,4
 2100 COLOUR131:CLS
 2110 MOVE 565,850
 2120 GCOL0,0
 2130 PLOT1,150,0:PLOT1,0,-80:PLOT1,-15
0,0:PLOT1,0,80
 2140 PLOT0,20,-40:PLOT1,110,0:PLOT0,-7
2,28:PLOT0,34,0
 2150 PLOT81,0,-53:PLOT0,-34,0:PLOT81,0
,53:PLOT0,-29,-48
 2160 GCOL0,1
 2170 PLOT65,0,0:PLOT65,4,0:PLOT65,0,4:
PLOT65,-4,0
 2180 COLOUR0:PRINTTAB(4,5)"DISC"'TAB(3
,7)"SECTOR"'TAB(3,9)"EDITOR"
 2190 PRINT''''''"   Drive? ";:A$=GET$:D
%=VAL(A$):A$=STR$(D%)
 2200 ENDPROC
 2210 DEFPROCcm(A$):$blc%=A$:X%=blc%MOD
256:Y%=blc%DIV256:CALL&FFF7:ENDPROC
```

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

WHICH BASIC?

Dear Sir,

Can you tell me whether the 1.2 Operating System ROM has Basic I or Basic II?

M. Childs

[The answer is: Neither. The Operating System and Basic are both supplied as a chip - a "Read Only Memory" or ROM, but they are quite separate. You could for example have a 1.2 Operating System and EITHER Basic I or Basic II.

The Operating System and the Basic chips work in similar ways (in providing a series of instructions in machine code for the 6502 microprocessor in the Beeb); but the former is mainly concerned with routine low level activities like putting characters on the screen, handling cassette SAVE/LOAD operations, handling keyboard entry etc. The Basic ROM on the other hand handles the running of Basic programs, and the interpreting of each Basic command. See the article 'Which Version?' in this issue for the way to discover whether you have Basic I or Basic II. Since Feb/March, production models of the Beeb have been supplied with O.S 1.2 and Basic II. D.E.G.]

TAPE RECORDER REVIEW

After reading reviews of cassette recorders in the April/May issue of BEEBUG MAG, I have to add a further item to your list, which I appreciate is necessarily limited. This is the Philips N2234.

The machine offers full auto-stop on record, play, fast forward or rewind. Aslo available are full cue and review facilities tape counter and tone control. The counter tone and volume controls are located on the top surface of the machine, input/output and remote ports are miniature jack sockets on the left hand side.

As a maintenance engineer working with a wide variety of electronics I appreciate a well designed and constructed piece of equipment and the Philips is by far the easiest recorder I have ever worked on. (I take all of my equipment apart on receipt!). I have used this recorder for six months and despite a Ø.1 O.S. have had no loading problems at all and that with no bug fix! The recorder is available from Curry's at around £29.

PS As a member of HM Forces I have absolutely no connection with either Philips or Curry.

T.S. Day

WORDWISE OFFER

How dare you! All those nice advertisers in Computer Magazines, the National Press, Sunday Colour Supplements, and not to mention the Radio Times, have spent a small fortune in persuading us all that when they say that we should wait for the answer to life, the universe, and everything (ie 28 days) they really mean it. In fact, waiting is so good for the soul - combining as it does all the virtues of patience, self denial, and struggle with the demons of doubt that the Coming of the Software will ever happen in our time - that it may be said with pardonable pride that these Good Purveyors have contributed much to the British Character.

So what do you think you are doing ? I make out my cheque, put it in an envelope with a letter requesting your special Wordwise Offer, and consign it to the mercies of the British Postal System on Friday. I then spend a happy weekend getting the creases out of my sackcloth and preparing a fresh batch of ashes, and settling down to a good session of meditation on the Awesome Significance of 28. My RAM over-floweth.

Confusion! Terror! The BEEBUG special Wordwise Offer arrives at my door on MONDAY !!

Is nothing sacred to you ? Have you no remorse ? I can only charitably believe that the excellence of your magazine has gone to your head. What other reason can there be for you to commit such heretical acts and stray so

far from the paths of the righteous ? For your transgressions, spool together 28 Ave Clives, then go Forth and do it again. CONGRATULATIONS BEEBUG !!!

Mike Simon

TORCH REVIEW

Mr K R Pinker of Schaverien & Co. (Members of the Stock Exchange), writes:

"..Whilst much of your comment (Torch Disc Pack Review BEEBUG vol 2 no 1) is quite fair, I received a BBC utilities disc when I purchased my Torch contrary to the impression that you gave. I understand the BBC Basic for the Torch is almost ready, and when this is available, taking only 16k of the 63k available, the user can use the high resolution mode with 47k of RAM available when using BBC Basic loaded under CPN.... To refer to the Z80 pack as "almost a Torch" is a great compliment as a Torch costs £2700. I am delighted with my disc pack and feel you could have put a greater emphasis on its more positive points".

Mr P Headland of Cambridge Computer Consultants Ltd. adds the following:

"Torch no longer suggest you replace the BBC Micro's internal power supply as the new switched mode power supplies are powerful enough to take the extra load of the Z80 card.... There is no track buffering under DFS and this is not altered by Torch. The lack of clicking from other drives is caused by a lack of head-load solenoids. The drives are designed to withstand repeated head loading with no significant wear.... I admit that CP/M software seems expensive to the hobbyist, but that is scarcely Torch's fault. Viewed from a more sensible perspective, the BBC Micro/Torch Z80 Disc Pack combination is a suprisingly cheap way of getting a very sophisticated and high-performance CP/M machine, and as such is really rather more intended for the professional user than the enthusiast."

[Quite a number of issues are raised here and it is worth commenting briefly on each. First, with respect to the Basic. At the time of the review there was not even a hint from Torch that such a package might be available. After a bit of spade-work we now know that a firm in Norwich has commissioned the writing of a CP/M compatible BBC Basic. It is not possible to say much more about this at the current time but the cost should be *around £110 for a* complete Basic Interpreter. This represents an appreciable extra expenditure. Also at the time of the review Torch were still saying that the power supply required changing. Apparently this was a 'blanket' statement which kept the instructions short! We are now informed, as you suggest, that if you have a 'switched-mode' power supply, you do not need to change it. Your comment about the Torch not using track buffering on disc accesses is in fact incorrect, as Torch system engineers will confirm. What may be useful to know however is that you can change the links in the disc drives to make the drives use the 'head & motor' system of operation, rather than the default 'head & select' mechanism of a standard Torch drive. This will make the drives act in a similar way to other drives under Acorn's DFS, (though on our review set this did not make a significant difference), but will cause an increased amount of wear and tear because of the amount of time the head spends on the disc. In terms of cost it may be o.k for businesses such as Consultancies and Members of the Stock Exchange to purchase such equipment (plus additional software) but only a very small prportion of our members are in that kind of a position. In relative terms the Torch system is remarkable value for money but in absolute terms the buyer who wants to keep using BBC Basic must eventually pay just over £1000 in total. I would also challenge the statement that the BBC Micro/Torch disc pack is a cheap way of getting a CP/M machine - there are in fact many CP/M micros on the market at a cheaper all inclusive price. Finally we should add that we have received several letters from Torch Disc Pack owners who expressed reservations about their purchase for the kind of reasons given in the review. C.N.O.]

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

LINE LISTING AFTER ERROR - P. Jollyman

If you use the following key definition, you will get a listing of the line signalled in the last ERROR statement, each time you press function key zero. This can be extremely useful for editing purposes. Just press f0 when an error occurs, and you will see the offending line. There are two modifications that you may wish to make

(1) Insert MO.7|M at the very start (i.e. after *KEY0). This will put the machine into mode 7 before printing the error line

(2) After STR$ERL insert +","

This allows you to continue from a listing that you have escaped from. It might be useful to put this version into a different key.

```
*KEY0 C$="LIST"+STR$ERL+CHR$13:A%=138:X%=0:F.L=1TOLENC$:Y%=ASC(MID$(C$,L)):CA.&FFF4:
N.|M
```

FAST VERIFY FOR WORDWISE - David Graham

A very quick verify operation for Wordwise files can be performed using the 'verify' suggested for Basic and other programs in BEEBUG v.1 no.3. To verify a file name TEXT, simply type:

```
*LOAD TEXT &8000
```

This does not check it against text held in memory, but it does the next best thing. Any loading faults will be signalled, and the test does not affect text in memory. This means that if it is performed immediately after saving a given file, you have the opportunity of re-saving it if the 'verify' indicates a fault. Note that the verify routine works very many times faster than the original data saving or loading operation, (Disc only!!). Of course it is always a good idea to keep back-up copies of all work.

CHANGING "TAB" CO-ORDINATES TO "MOVE" PARAMETERS - A.E. Wilmshurst

In the series 1 operating systems, the TAB command does not function when a VDU5 has been issued. Many programs have been written on O.S. 0.1 assuming that TAB does work. Converting these programs consists of changing the TAB statements to MOVE (it was not supposed to operate on 0.1 but there must have been an "unbug"). The conversion from text co-ordinates to graphics points is easily performed. Where you see TAB(X,Y) use MOVE X*factor,1020-Y*32. Where factor is 16 for mode 0, 32 for modes 1 and 4, and 64 for modes 2 and 5. A program written this way will work on all current operating systems (unless something else goes wrong!).

BASIC PROGRAM EXECUTION ADDRESS - Richard Porter

To run a Basic program from machine code, the following is all that is required (Basic I):

```
JMP &BD2C
```

MORE FX CALLS

A member called 'Mac' would like to add a few more calls to the ever increasing list of *FX commands, and so would the editorial team:

*FX 200 Has various effects depending on the value of the second argument. Those known of are:

*FX200,0 - turns Escape key ON

*FX200,1 - turns Escape key OFF (This is different from *FX220, which re-programs the Escape key. *FX220,0 programs it to CTRL @)

*FX200,2 - wipes out memory after 'Break' completely destroying programs in memory. You can of course achieve this by programming function key 10.

*FX201 Switches on/off the entire keyboard (except Break).

*FX201,0 - turns keyboard ON

*FX201,1 - turns keyboard OFF

Program tested on O.S. 0·1 and 1·2

ANAGRAMS (16k)

by Mike Case

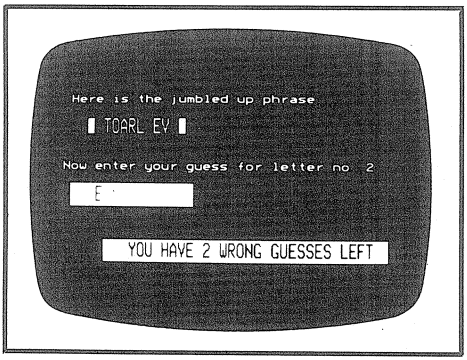
ANAGRAMS is rather different from most games. It requires a degree of thought on the part of the player - and moreover it has no 'fire button' - but do not let this put you off. The player is presented with anagrams which he must solve, each of which has at least one correct solution.

SARAMNAG

The list of words used is held in DATA statements from line 1810 onwards. The words are terminated by 999 to mark the end of the list, and new words can be inserted in fresh data statements, providing you make sure that 999 is the last item of data. The maximum number of words currently allowed is 200, set at line 40. Change vocab$(200) to a larger value if required, though if this becomes too large you will need a 32k machine.

NAMAGRAS

To make the game suitable for children, either remove the longer words and replace them with simpler ones, or alter the section at line 220 to filter out long words, e.g. insert 255 IF length>4 THEN GOTO 220 (you can also filter out the shorter words by a similar process). If the children are very young you may wish to put the words into lower case, as they will probably not be familiar with upper case text; though of course you cannot easily change the keys themselves! Note that some of the anagrams displayed may have more than one solution. The program will only accept ONE of these. For this reason we have edited out a considerable number of the 3 and 4 letter words intially supplied. Note that lines 1710 to 1790 are data statements used to generate Teletext Colour codes for the display. This technique has been used to facilitate keyboard entry of the program.




```
  10 REM ANAGRAMS  VERSION  1A
  20 REM BY MIKE CASE ; SEPTEMBER 1982
  30 MODE 7
  40 DIM CL$(9),vocab$(200),guess$(20)
,jumbled$(20)
  50 PROCSETCOL
  60 REM DISABLE CURSOR
  70 VDU23;8202;0;0;0;
  80 W%=0:I%=0
  90 CLS
 100 PROCTITLE
 110 FOR wait=1 TO 1000:NEXTwait
 120 CLS
 130 FORJ%=0TO1:PRINTTAB(7,2+J%) CL$(1
);"SCORE IS ";W%;" OUT OF ";I%;SPC(2);C
HR$156:NEXT
 140 REM Set random number generator t
o random start
 150 X=RND(-TIME)
 160 RESTORE 1810
 170 REM READ IN THE WORDS
 180 vocab=0
 190 REPEAT vocab=vocab+1
 200 READ vocab$(vocab)
 210 UNTIL vocab$(vocab)="999":NUM=voc
ab-1
 220 REM Pick a random word
 230 random=RND(NUM)
 240 phrase$=vocab$(random)
 250 length=LEN(phrase$)
 260 guess=1
 270 right=0:wrong=0
 280 PROCSHUFFLE
 290 PRINT''SPC(7);"The word you have
to find is "
 300 PRINTSPC(7);length;" letters long
, you have to"
 310 PRINTSPC(7);"guess each letter in
 turn,"
```

```
  320 PRINTSPC(7);"starting at the firs
t one. "
  330 PRINT'':FORJ%=ØTO1:PRINTSPC(2);CL
$(8);"PRESS THE SPACE BAR TO START";SPC
(2);CHR$156:NEXT
  340 REPEAT UNTIL GET=32
  350 CLS
  360 FORJ%=ØTO1:PRINT'CHR$141;CHR$130;
" Length = ";length;:NEXT
  370 PRINTTAB(2,5)"Here is the jumbled
 up phrase"
  380 FORJ%=ØTO1:PRINTTAB(2,7+J%);CL$(7
);shuffled$;CHR$134;CHR$255:NEXT
  390 PRINTTAB(1,11)"Now enter your gue
ss for letter no. ";guess
  400 guess$(guess)=GET$
  410 REM to prevent guess of letter no
t in word
  420 PROCSILLY
  430 IF silly=Ø THEN GOTO400
  440 PROCCHECK
  450 IF check=Ø THEN PROCNOISE:wrong=w
rong+1
  460 IF check=1 THEN PROCYES:guess=gue
ss+1
  470 REM Set number of wrong guesses a
llowed to INT(half total letters)
  480 IF (INT(length/2))-wrong=1 THEN F
ORJ%=ØTO1:PRINTTAB(4,18+J%);CL$(2);"THI
S IS YOUR VERY LAST CHANCE ":NEXT:GOTO5
ØØ
  490 FORJ%=ØTO1:PRINTTAB(4,18+J%);CL$(
2);"YOU HAVE ";(INT(length/2))-wrong;"
WRONG GUESSES LEFT ":NEXT
  500 IF check=1 THEN right=right+1
  510 end$=""
  520 IF wrong=INT(length/2) THEN PROCA
NSWER
  530 IF right=length THEN PROCWIN
  540 IF end$="" THEN 390
  550 IF end$="N" OR end$="n" THEN MODE
7:FORJ%=ØTO1:PRINTTAB(8,8+J%);CL$(3);"B
 Y E - B Y E   ";CHR$156:NEXT:END
  560 IF end$="Y" OR end$="y" THEN GOTO
120
  570
  580 DEFPROCCHECK
  590 check=Ø
  600 IF guess$(guess)=MID$(phrase$,gue
ss,1) THEN check=1
  610 IF check=1 AND guess=1 THEN FORJ%
=ØTO1:PRINTTAB(1,13+J%);CL$(4);SPC(leng
th+5);CHR$156:NEXT
  620 IF check=1 THEN FORJ%=ØTO1:PRINTT
AB(3,13+J%);CHR$141;CHR$132;LEFT$(phras
e$,guess);CHR$158:NEXT:PRINTTAB(4,9)SPC
(length+3):PROCDELETE
  630 IF check=Ø THEN PROCUNDERLINE
  640 ENDPROC
  650
  660 DEFPROCNOISE
  670 SOUND 3,-15,100,5
  680 SOUND 2,-15,103,5
  690 SOUND Ø,-12,20,5
  700 ENDPROC
  710
  720 DEFPROCYES
  730 ENVELOPE 1,1,2,-2,5,30,25,15,50,Ø
,Ø,255,120,Ø
  740 SOUND 1,1,80,5
  750 SOUND 1,1,100,5
  760 ENDPROC
  770
  780 DEFPROCANSWER
  790 I%=I%+length:W%=W%+right
  800 shuffled$=""
  810 FOR times=1 TO length
  820 shuffled$=shuffled$+jumbled$(time
s)
  830 NEXTtimes
  840 CLS
  850 PRINTTAB(5,1)"The jumbled phrase
was:-"
  860 FORJ%=ØTO1:PRINTTAB(5,3+J%);CL$(5
);shuffled$;SPC(3);CHR$156:NEXT
  870 PRINTTAB(4,7)"The original phrase
 was :-"
  880 FORJ%=ØTO1:PRINTTAB(5,9+J%);CL$(6
);phrase$;SPC(2);CHR$156:NEXT
  890 PRINT''SPC(4);"Another go (Y or N
) ?"
  900 end$=GET$
  910 IF end$<>"y"AND end$<>"Y" AND end
$<>"n"AND end$<>"N" THEN 900
  920 ENDPROC
  930
  940 DEFPROCTITLE
  950 PRINT'''';:FORJ%=ØTO1:PRINTCHR$130
;CHR$141;SPC(8);"A N A G R A M":NEXT
  960 PRINTTAB(6,1);CHR$146;STRING$(20,
"s")
  970 PRINTTAB(5,1);CHR$146;"7";TAB(27,
1);"k"
  980 PRINTTAB(27,6);"z"
  990 PRINTTAB(6,6);CHR$146;STRING$(20,
"s")
 1000 PRINTTAB(5,6);CHR$146;"u"
 1010 FOR I=2 TO 5
 1020 PRINTTAB(6,I);CHR$146;"5"
 1030 PRINTTAB(5,I);CHR$146;"5"
 1040 PRINTTAB(26,I);CHR$146;"j"
 1050 PRINTTAB(25,I);CHR$146;"j"
 1060 NEXT
 1070 ENDPROC
 1080
 1090 DEFPROCSHUFFLE
 1100 IF LEFT$(phrase$,1)=" " THEN phra
se$=RIGHT$(phrase$,(LENphrase$-1)):leng
th=length-1
 1110 IF LEFT$(phrase$,1)=" " THEN 1100
 1120 space=Ø
 1130 new$=phrase$
```

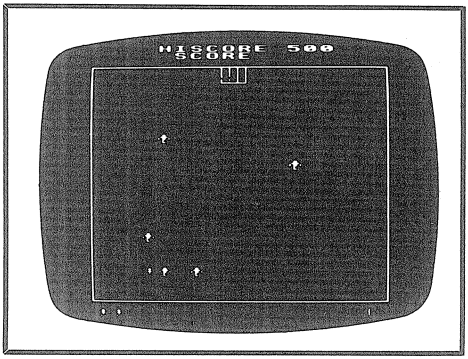
```
 1140 FOR times=1 TO length
 1150 IF times=length X=1:GOTO1170
 1160 X=RND(LEN(new$))
 1170 jumbled$(times)=MID$(new$,X,1)
 1180 new$=LEFT$(new$,(X-1))+RIGHT$(new
$,(LEN new$-X))
 1190 NEXTtimes
 1200 shuffled$=""
 1210 FOR times=1 TO length
 1220 shuffled$=shuffled$+jumbled$(time
s)
 1230 IF jumbled$(times)=" " THEN  spac
e=space+1
 1240 NEXTtimes
 1250 ENDPROC
 1260
 1270 DEFPROCWIN
 1280 PROCYES:PROCYES
 1290 W%=W%+right:I%=I%+length
 1300 CLS
 1310 IF wrong>1 THEN FORJ%=0TO1:PRINTT
AB(0,5+J%);CL$(9);"YOU WIN WITH ONLY ";
wrong;" WRONG GUESSES":NEXT
 1320 IF wrong=1 THEN FORJ%=0TO1:PRINTT
AB(0,5+J%);CL$(9);"YOU WIN IN ONLY ONE
WRONG GUESS":NEXT
 1330 IF wrong=0 THEN FORJ%=0TO1:PRINTT
AB(0,5+J%);CL$(9);"WELL DONE !!  NO WRO
NG GUESSES":NEXT
 1340 PRINT''''"  DO YOU WANT ANOTHER GO
 ?"
 1350 end$=GET$
 1360 IF end$<>"Y"AND end$<>"N" AND end
$<>"n" AND end$<>"y" THEN 1350
 1370 ENDPROC
 1380
 1390 DEFPROCUNDERLINE
 1400 find=0
 1410 REPEAT find=find+1
 1420 UNTIL guess$(guess)=MID$(shuffled
$,find,1)
 1430 PRINTTAB(4,9);SPC(20)
 1440 PRINTTAB((4+find),9);CHR$145;CHR$
96
 1450 ENDPROC
 1460
 1470 DEFPROCDELETE
 1480 find=0
 1490 REPEAT find=find+1
 1500 UNTIL guess$(guess)=MID$(shuffled
$,find,1)
 1510 shuffled$=LEFT$(shuffled$,find-1)
+" "+RIGHT$(shuffled$,(LENshuffled$-fin
d))
 1520 FORJ%=0TO1:PRINTTAB(2,7+J%)CL$(7)
;shuffled$;CHR$134;CHR$255:NEXT
 1530 ENDPROC
 1540
 1550 DEFPROCSILLY
 1560 silly=0
 1570 FOR find=1 TO length
 1580 IF guess$(guess)=MID$(shuffled$,f
ind,1) THEN silly=1
 1590 NEXTfind
 1600 ENDPROC
 1610
 1620 DEFPROCSETCOL
 1630 RESTORE 1710
 1640 FORI%=1 TO 9
 1650 REPEAT
 1660 READ C:IFC<>0 THEN CL$(I%)=CL$(I%
)+CHR$(C)
 1670 UNTIL C=0
 1680 NEXT
 1690 ENDPROC
 1700
 1710 DATA132,157,141,131,0
 1720 DATA141,135,157,136,129,0
 1730 DATA131,157,141,132,0
 1740 DATA135,157,141,0
 1750 DATA129,157,141,135,0
 1760 DATA135,157,141,129,0
 1770 DATA141,134,255,131,0
 1780 DATA147,141,157,145,0
 1790 DATA135,157,141,132,136,0
 1800
 1810 DATA HOUSE,COMPLAINT,OFFENCE,PROT
RUDE,PROPORTIONS
 1820 DATA ACCESSION,ADJACENT,ADEQUATE,
ADHESIVE,BENEVOLENT,CONVERT,GRAIN
 1830 DATA GROOM,PESSIMISM,PERFECT,HOLD
,BRACELET,CONVENIENCE
 1840 DATA DATE,CREOSOTE,CRATER,CRAVE,C
RAB,LEAST,LEFT
 1850 DATA TREASURE,TRADE,DART,CORNET,C
RANE,CARROT,LAPEL,CRADLE
 1860 DATA CONTAIN,TRIP,PRESUME,PLANE,K
NEEL,CABINET
 1870 DATA DIVIDE,DRIED,IDEA
 1880 DATA PARTITION,AVOIDANCE,ENDURANC
E,ELEPHANT,SLIPPER,SLIDE
 1890 DATA GRAPES,PRIDE,BOTHER,BRAIN,BR
AID,BEARD
 1900 DATA AGGRIEVED,VERTIGO,ELEVATOR,E
XCAVATOR,AVIATOR,UMBRELLA,UNDER
 1910 DATA SUGAR,STARVE,STERN,IF,IMAGE,
INFECT
 1920 DATA UNDERSTAND,LOVERS,SLAM,APPLI
CATION,EVENTUALITY
 1930 DATA QUEEN,HOT,FEAR,PRAM,REAM,CAR
VE,CROCODILE,APPLE
 1940 DATA EXACERBATE,EXCOMMUNICATION,D
RONE,GRAPE,GALE,FLAG
 1950 DATA HEAP,EPIC,EVIL,WHAT,THING,999
```

Program tested on O.S. 0.1 and 1.2

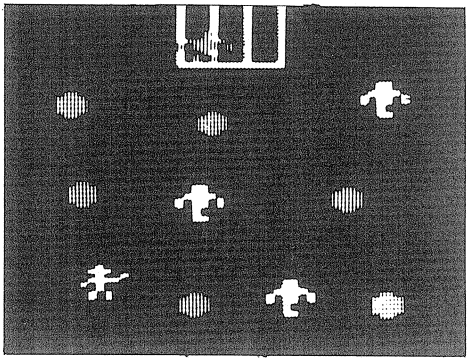
ROBOT ATTACK (32k)

by Edward Hung

This program is an implementation of the classic Zombies game, which has been written on many computers. You control a man who is chased by a number of Robots. You must dodge them, and at the same time lure them into variously placed mines. You must also keep well



clear of the swift and purposeful Marvin. From time to time he will burst through the bars of his cage to pursue you - but you do get a warning - the cage bars will flash if this is imminent. ROBOT ATTACK is pleasantly fast and fun to play , requiring a nice balance of dextrous skill and strategy. The game has a number of nice optional extras - a sound on/off option, a high score table, instructions and so on. The program is in colour and requires a 32k machine. Disc users should set PAGE to &1100 before typing or loading in (i.e. type PAGE=&1100 <return>), and should avoid pressing Break. For this reason, disc users may wish to delete line 2070 to allow Escape to function.



The program is entirely in Basic. The single CALL command is for OSWORD call zero to input the player's name. This technique is extremely useful as INPUT is rather unsatisfactory for this purpose, allowing any character to be typed. It is also much shorter than trying to write a Basic routine to input a line in this way. The relevant lines are 1830 to 1860 which produce a string N$. The control block is set up to allow a maximum of 20 characters, each in the range ASCII 32 to 128.

```
  10 REM *************************
  20 REM **     ROBOT ATTACK     **
  30 REM **    Written for the   **
  40 REM **  BBC Micro Model B   **
  50 REM **    By Edward Hung    **
  60 REM **      VERSION 1A      **
  70 REM *************************
  80 PROCSETUP
  90 ON ERROR GOTO 2060
 100 MODE7:PROCTITLEPAGE
 110 MODE2
 120 SP%=SP%-50
 130 PROCPLOTOBSTACLES
 140 REPEAT
 150 I%=I%+1
 160 PROCCOWBOY
 170 PROCROBOT
 180 IFGKL%=1 AND GA%=1 AND DD%=0 AND
RND(1)>.3 PROCMARVIN
 190 IFI%>5 I%=0
 200 IFS%>=OS% OS%=OS%+1500:LV%=LV%+1:
PROCLV:IFSND%=1 SOUND1,1,100,1
 210 UNTIL DP%=6 OR DD%=1
 220 IFDD%=0 SH%=SH%+1:DP%=0:GOTO120
 230 LV%=LV%-1:DD%=0:IFLV%>0 GOTO130
 240 PROCDELAY(3500)
 250 PROCSCORE:PROCRESET:PROCDISPLAYHI
SCORE
 260 GOTO110
 270 DEFPROCTITLEPAGE
 280 FORI%=8TO9:VDU31,11,I%,141,129:PR
INT"ROBOT ATTACK":NEXT
 290 PRINT'TAB(11);CHR$(&81);"By";CHR$
(&82)+"Edward Hung"''"Do you wish to be
 briefed on the mission"TAB(11);CHR$(&8
8);CHR$(&86);"PRESS Y OR N"
```

```
  300 A$=GET$:IFA$="Y" PROCINSTRUCTION
ELSE IFA$<>"N" THEN300
  310 ENDPROC
  320 DEFPROCINSTRUCTION
  330 VDU12,31,5,1:PRINT"ROBOT ATTACK I
NTRUCTIONS"
  340  PRINT'"      You are the last sur
vivor of the    Human  Race.  You are
being  persued"
  350 FORT%=1TO13:READA,A$:PRINTTAB(A)A
$:NEXT
  360 PRINT'TAB(8)"SPACE BAR TO HYPERSP
ACE"''TAB(9);CHR$(&85);CHR$(&88);"ANY
KEY TO START"
  370 DATA0,"by a hostile group of Bolo
 Bolo Robots,",1,"whose only orders are
 to kill you by",13,"ramming you.",2,"O
ccasionally, Mean MARVIN,King of",4,"
Boloansappears to help them.",2,"You m
ust kill them by luring them"
  380 DATA2,"onto mineswhich you yours
elf must",13,"not walk onto.",3,"CONTRO
LS:",16,"Z   LEFT",16,"X   RIGHT",16,":
  UP",16,"/   DOWN"
  390 *FX15,0
  400 A=GET
  410 ENDPROC
  420 DEFPROCSETUP
  430 ENVELOPE1,1,0,0,0,0,0,0,0,0,0,-1,
126,0
  440 ENVELOPE2,129,2,4,6,28,14,7,0,0,0
,-80,80,80
  450 ENVELOPE3,1,4,-4,4,10,20,10,127,0
,0,-5,126,126
  460 VDU23,250,8,28,9,30,40,8,20,20
  470 VDU23,251,16,56,144,120,20,16,40,
40
  480 VDU23,252,24,24,60,90,90,16,16,24
  490 VDU23,253,24,16,124,186,186,16,40
,40
  500 VDU23,254;24,60,60,60,60,24;
  510 VDU23,255;146,84;198;84,146
  520 VDU23,224,160,84,184,16,8,5,2;
  530 VDU23,225;0;0,9,81,127;
  540 VDU23,226,5,42,29,16,32,192,64;
  550 VDU23,227;0;0,72,69,127;
  560 VDU23,228,1,1,1,1,1,1,1,1
  570 VDU23,229,0,6,30,62,30;0;
  580 DIMX%(6),Y%(6),S%(10),N$(10)
  590 SND%=1:*FX4,1
  600 PROCSCINIT
  610 PROCRESET
  620 ENDPROC
  630 DEFPROCSCINIT
  640 FORI%=1TO10:S%(I%)=550-I%*50:N$(I
%)="The Robots of Zaexon":NEXT
  650 ENDPROC
  660 DEFPROCLV
  670 VDU17,3,31,1,31:PRINTSPC(14);TAB(
1,31)STRING$(LV%-1,CHR$250);
  680 ENDPROC
  690 DEFPROCRESET
  700 S%=0:SC%=0:OS%=1500:SP%=400:SH%=1
:LV%=3:F%=0
  710 ENDPROC
  720 DEFPROCRND1
  730 A%=RND(18):B%=RND(25)+4
  740 ENDPROC
  750 DEFPROCRND2
  760 X%(I%)=RND(18):Y%(I%)=RND(25)+4
  770 ENDPROC
  780 DEFPROCRND3
  790 X%=RND(18):Y%=RND(25)+4
  800 ENDPROC
  810 DEFPROCPLOTOBSTACLES
  820 VDU23;8202;0;0;0;28,1,29,18,3,12,
26,19,15,2;0;
  830 GCOL0,7:MOVE60,60:DRAW1219,60:DRA
W1219,930:DRAW60,930:DRAW60,60:GCOL0,15
:FORI%=576TO704STEP33:MOVEI%,926:DRAWI%
,866:NEXT:DRAW576,866
  840 FORT%=1TO15:VDU17,1,31,RND(18),RN
D(24)+5,254:NEXT
  850 FORI%=1TO6
  860 REPEATPROCRND2:UNTILFNPOINT=0
  870 VDU17,6,31,X%(I%),Y%(I%),252:NEXT
  880 VDU5,18,0,2,25,4,576;1024-4*32;25
3,4,17,5,31,5,0
  890 PRINT"HISCORE ";S%(1)TAB(6,1)"SCO
RE ";S%
  900 I%=0:DD%=0:GKL%=-1:DP%=0:D%=250:G
A%=0:FL%=0:C%=0
  910 PROCLV
  920 LE%=SH%MOD5
  930 IFSH%/5=SH%DIV5 LE%=5
  940 IFLE%=1 VDU4:PRINTTAB(14,31);SPC(
4);
  950 VDU5,18,0,1,25,4,(18-LE%)*64;1024
-31*32;
  960 IFSH%>30 SH%=1
  970 IFSH%<31 A=7:IFSH%<26 A=6:IFSH%<2
1 A=5:IFSH%<16 A=2:IFSH%<11 A=4:IFSH%<6
 A=1
  980 GCOL0,A:PRINTSTRING$(LE%,CHR$(229
))
  990 GCOL0,3:MOVE(18-LE%)*64,1024-31*3
2:PRINTTAB(18-LE%,31)STRING$(LE%,CHR$(2
28))
 1000 REPEATPROCRND1:UNTILFNPNT(A%,B%)=0
 1010 a%=A%:b%=B%
 1020 VDU4,17,12,31,A%,B%,250
 1030 FORT%=1TO10000:NEXT
 1040 VDU17,3,31,A%,B%,250
 1050 ENDPROC
 1060 DEFPROCCOWBOY
 1070 IFINKEY-98   A%=A%-1:F%=1:D%=251
 1080 IFINKEY-67   A%=A%+1:F%=1:D%=250
 1090 IFINKEY-105 B%=B%+1:F%=1
 1100 IFINKEY-73   B%=B%-1:F%=1
 1110 IFINKEY-99   H%=1:F%=1:REPEATPROCR
ND1:UNTILFNPNT(A%,B%)<>6 AND FNPNT(A%,B
%)<>2:IFSND%=1 SOUND17,2,50,10
```

```
1120 IFF%=0 ENDPROC
1130 A%=A%+(A%>18)-(A%<1):B%=B%+(B%>29
)-(B%<3)
1140 IFFNPNT(A%,B%)=1 PROCPIT(a%,b%,A%
,B%):DD%=1:ENDPROC
1150 IFFNPNT(A%,B%)=6 OR FNPNT(A%,B%)=
2 PROCDEAD:DD%=1:ENDPROC
1160 IFFNPNT(A%,B%)<>0 A%=a%:B%=b%:END
PROC
1170 PRINTTAB(a%,b%)" ":IFH%=1H%=0:PRO
CDELAY(700)
1180 VDU17,3,31,A%,B%,D%
1190 a%=A%:b%=B%:F%=0
1200 FORT%=1TOSP%:NEXT
1210 ENDPROC
1220 DEFPROCROBOT
1230 IFDD%=1 ENDPROC
1240 IFX%(I%)=0 AND Y%(I%)=0 ENDPROC
1250 OX%=X%(I%):OY%=Y%(I%)
1260 X%(I%)=X%(I%)+(X%(I%)>A%)-(X%(I%)
<A%)
1270 Y%(I%)=Y%(I%)+(Y%(I%)>B%)-(Y%(I%)
<B%)
1280 IFFNPOINT=6 OR FNPOINT=2 REPEAT P
ROCRND2:UNTIL FNPOINT=0:IFSND%=1 SOUND1
,2,50,10
1290 IFFNPOINT=1 PROCPIT(OX%,OY%,X%(I%
),Y%(I%)):X%(I%)=0:Y%(I%)=0:DP%=DP%+1:P
ROCMS(10):ENDPROC
1300 IFFNPOINT=3 PROCDEAD:DD%=1:ENDPROC
1310 IFFNPOINT<>0 Y%(I%)=Y%(I%)+2
1320 FORT%=1TOSP%:NEXT
1330 IFSND%=1 SOUND3,-15,10,1
1340 PRINTTAB(OX%,OY%);SPC(1):VDU17,6,
31,X%(I%),Y%(I%),252
1350 IFRND(1)>0.95 AND FL%=0:GKL%=-GKL
%:C%=GKL%:IFGKL%=1 X%=9:Y%=5:FL%=1:J%=0
:GA%=0:VDU19,15,14;0;:IFSND%=1 SOUND1,-
15,50,1
1360 IFC%=-1 VDU5,18,3,2,25,4,9*64;1024-4*32;253,4:PRINTTAB(X%,Y%)SPC(1):C%=0
:IFSND%=1 SOUND1,-15,200,1
1370 IFFL%=1 J%=J%+1:IFJ%=10 VDU5,18,3
,2,25,4,9*64;1024-4*32;253,4,19,15,2;0;
:FL%=0:GA%=1:IFSND%=1 SOUND1,-15,200,1
1380 ENDPROC
1390 DEFPROCMARVIN
1400 XX%=X%:YY%=Y%
1410 X%=X%+(X%>A%)-(X%<A%):Y%=Y%+(Y%>B
%)-(Y%<B%)
1420 IFFNPNT(X%,Y%)=15 Y%=Y%+2
1430 IFFNPNT(X%,Y%)=1 PROCPIT(XX%,YY%,
X%,Y%):X%=0:Y%=0:GKL%=-1:VDU5,18,3,2,25
,4,576;1024-4*32;253,4:PROCMS(100):ENDP
ROC
1440 IFFNPNT(X%,Y%)=3 PROCDEAD:DD%=1:E
NDPROC
1450 IFFNPNT(X%,Y%)=6 REPEATPROCRND3:U
NTIL FNPNT(X%,Y%)=0:IFSND%=1 SOUND3,2,5
0,10
1460 IFSND%=1 SOUND2,3,50,1
1470 PRINTTAB(XX%,YY%);SPC(1):VDU17,2,
31,X%,Y%,253
1480 ENDPROC
1490 DEFPROCPIT(a,b,A,B)
1500 PRINTTAB(a,b)" "
1510 IFSND%=1 SOUND0,1,5,1
1520 VDU17,3,31,A,B,255:FORL=1TO500:NE
XT:PRINTTAB(A,B);SPC(1)
1530 ENDPROC
1540 DEFPROCDEAD
1550 IFD%=250 D%=224 ELSE D%=226
1560 VDU17,3,31,a%,b%,D%:FORL=1TO500:N
EXT:VDU31,a%,b%,D%+1
1570 IFSND%=1 SOUND0,1,5,1
1580 PROCDELAY(700)
1590 IFSND%=1 FORQ%=1TO5:FORL%=255TO20
0STEP-1:SOUND17,-15,L%,1:NEXT,
1600 IFSND%=0 PROCDELAY(2000)
1610 ENDPROC
1620 DEFPROCDELAY(T%)
1630 FORT=1TOT%:NEXT
1640 ENDPROC
1650 DEFPROCMS(S)
1660 S%=S%+S:VDU17,8,31,12,1:PRINT;S%
1670 ENDPROC
1680 DEFPROCSCORE
1690 FORI%=1TO10
1700 IFS%>=S%(I%) AND SC%=0 SC%=I%
1710 NEXT
1720 IFSC%=0 ENDPROC
1730 PROCENTERNAME
1740 FOR P%=10 TO SC%+1 STEP-1:S%(P%)=
S%(P%-1):N$(P%)=N$(P%-1):NEXT
1750 N$(SC%)=N$
1760 S%(SC%)=S%
1770 ENDPROC
1780 DEFPROCENTERNAME
1790 VDU12,17,2:PRINTTAB(0,6)"Your sco
re is in the      Top Ten"
1800 VDU17,6:PRINT" Please enter your"
;SPC(9);name:"
1810 *FX15,0
1820 VDU23;10,224;0;0;0;31,0,13;17,3
1830 X%=&80:Y%=&A:A%=0
1840 !&A80=&A00:?&A82=20:?&A83=32:?&A8
4=128
1850 CALL&FFF1
1860 N$=$&A00
1870 ENDPROC
1880 DEFPROCDISPLAYHISCORE
1890 VDU22,1,23;8202;0;0;0;19,3,12;0;:
PRINTTAB(15)"Highscores"'':VDU19,1,3;0;
17,1
1900 FORI%=1TO10:IFI%=10AA$=""ELSEAA$=
" "
1910 BB$=".....":IFS%(I%)<999BB$="....
.":IFS%(I%)<99BB$="......."
1920 PRINTTAB(3)AA$;I%;BB$;S%(I%);"...
";N$(I%)'
1930 NEXT
```

```
 1940 VDU19,2,15;0;17,1:PRINTTAB(8)"C t
o close sound channel"'TAB(8)"O to open
 sound channel"''TAB(11)"Sound Channel
"':COLOUR2:PRINTTAB(11)"Any key to star
t":COLOUR3:PRINT'TAB(7)"Or @ to clear s
core table"
 1950 COLOUR3:IFSND%=1 PRINTTAB(25,26)"
on. "ELSEPRINTTAB(25,26)"off."
 1960 *FX15,0
 1970 A=GET
 1980 IFA=64 PROCSCINIT:GOTO1890
 1990 IFA=67 OR A=99 PRINTTAB(25,26)"of
f.":SND%=0:GOTO1970
 2000 IFA=79 OR A=111 PRINTTAB(25,26)"o
n. ":SND%=1:GOTO1970
 2010 ENDPROC
 2020 DEFFNPOINT
 2030 =FNPNT(X%(I%),Y%(I%))
 2040 DEFFNPNT(U%,V%)
 2050 =POINT(U%*64+32,1010-V%*32)
 2060 ON ERROR OFF
 2070 IF ERR=17 THEN PROCRESET:GOTO 90
 2080 MODE7
 2090 *FX4,0
 2100 REPORT:PRINT" at line ";ERL
 2110 END
```

JOYSTICKS FOR ALIENS

by P. J. G. Butler

Ben Heller's ALIENS (BEEBUG Oct.'82) is such a success in our household that I began to fear for my keyboard, particularly with some of my children's heavy fingered friends. So when my BBC Joysticks arrived at last, I modified the program to use Joystick 1 to control the laser base position and fire, and also made a few minor modifications to the rest of the game, all listed here. It will also be necessary to modify the instructions as appropriate.

```
  45 ON ERROR MODE7:REPORT:
     PRINT" at line ";ERL:END
 225 R=RND(-TIME)
 440 BX% = ADVAL(1)/1500+1
 445 IF BX% = B1% ENDPROC
 450 IF BX% <1 BX%=1
 455 IF BX% >37 BX%=37
 495 BULL% = -(ADVAL(0) AND 1)
 500 IF YB%=0 AND NOT BULL% ENDPROC
 840 IF A1$="N" OR A1$="n" MODE7:END
```

A complete copy of the new version of Aliens appears on this month's magazine cassette. Note that it will not work without joysticks.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

*FX 3 (New O.S) - I. Foulkes, T.G.E. Bromilow, Matthew Rapier

As we mentioned last month (BEEBUG Vol.2 No.2 p.30). This call does not work as described in the User Guide page 422. The bits for RS423 and screen work correctly, but the printer bit doesn't. The main use of this command seems to be to select printing without screen display; and VDU2,21 is much the easiest way to replace the *FX3,2 call. However *FX3 has other capabilities. If bit 2 is set and bit 3 is clear (*FX3,4) then no printing can occur even if CTRL-B is pressed. To get printout bit 2 must be clear, and setting bit 3 will force printout without using CTRL-B. Another curious feature is bit 6. If this is set, the printer is disabled, but if it is clear and VDU21 is used to turn off the display then linefeeds (only) are sent out to the printer, if the printer ignore character is set to 0, this happens with RS423 and Centronics interfaces. To summarise:

bit 0 enables RS423
bit 1 disables Screen
bit 2 disables Printer
bit 3 forces Printer if CTRL-B not used
bit 4 disables Spooling
bit 6 disables Printer, also stops VDU21 emitting linefeeds to printer.

If you use RS423 try *FX3,9 with the printer set to be RS423.

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.

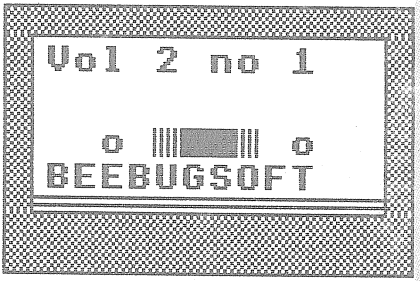
Editorial Address
BEEBUG
PO Box 50
St Albans
Herts
AL1 2AR

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 MAGAZINE 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 Matthew Rapier, John Yale, Adrian Calcraft, Tim Powys-Lybbe, Graham Greatrix and David Tall 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

Vol 2 no 1

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 GP100 SCREEN DUMP

Vol 2 no 3
programs include:

Balloons
Beeb at work
Speech programs
3-D Rotation Updated
Bad Program Lister
Files examples
Disc Sector Editor
Anagrams
Robot Attack
Aliens with Joysticks
PLUS A FULL LISTING OF RESCUE
SEIKOSHA GP250X DUMP
AND AN AUDIO REPRODUCTION OF THE VOCABULARY OF SPEECH PHROM A.

For ordering information see BEEBUGSOFT advertisment at the back of this month's magazine supplement.

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