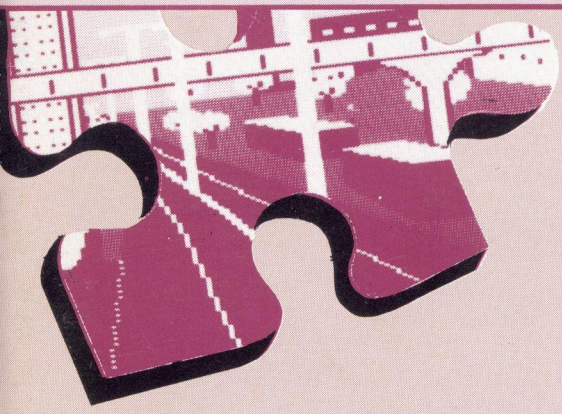
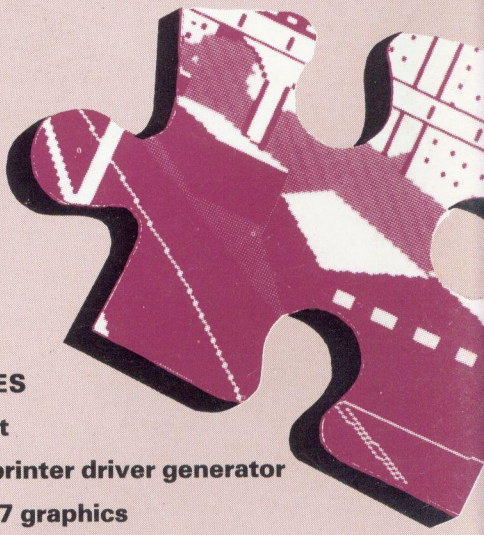
£1.20

BEEBUG

for the BBC micro

Vol 4 No 3 JULY 1985



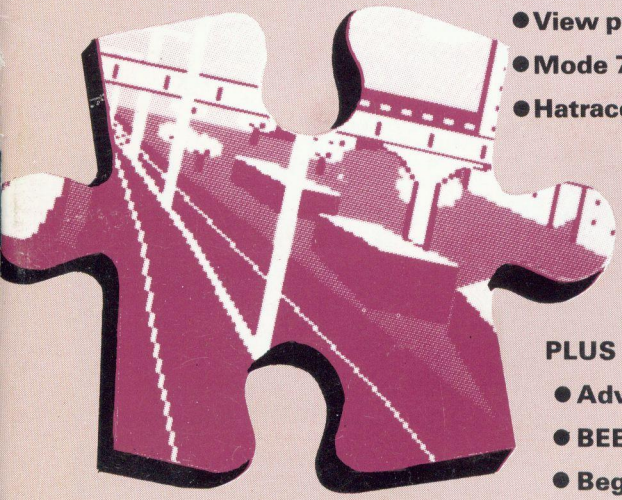


FEATURES

- 3D text
- View printer driver generator
- Mode 7 graphics
- Hatrace

REVIEWS

- Oxford Pascal
- Basic compiler
- Games
- Graphics tablets



PLUS

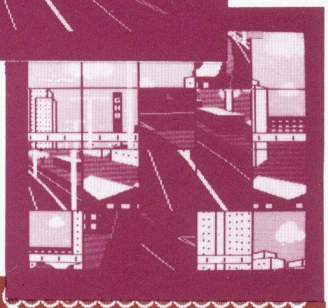
- Adventure Games
- BEEBUG Workshop
- Beginners start here
- And much more

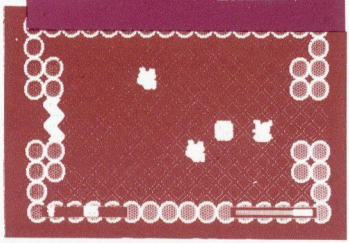


JIGSAW

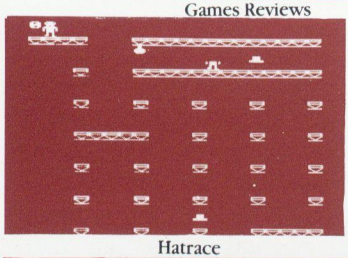


Jigsaw

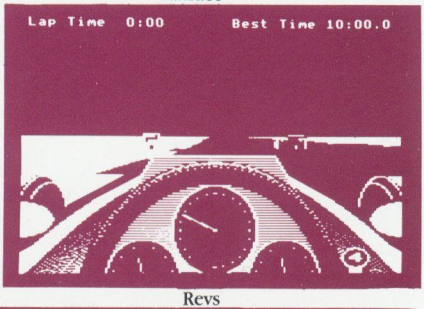




Games Reviews



Hatrace



Revs

BEEBUG

GENERAL CONTENTS

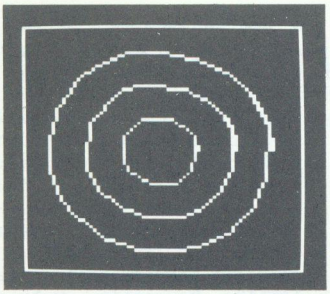
3 Editorial Jottings
4 Postbag
5 News
6 3D Text
9 Oxford Pascal Reviewed
11 Revs from Acornsoft
12 Jigsaw
16 Graphics Tablets Reviewed
18 Games Reviews
22 Letter Quality Printer Reviewed
23 Beginners Start Here
More about Logic
26 Generating a Printer Driver for View
31 Single Drive Disc Back-up
32 Accelerator
Computer Concepts Basic Compiler
34 Adventure Games
36 Graphics Commands for Mode 7
40 BEEBUG Workshop
Fast Circle Drawing
42 Book Reviews
44 EQU Functions for Basic I
45 Software Compatibility on the B+
46 Hatrace

PROGRAMS

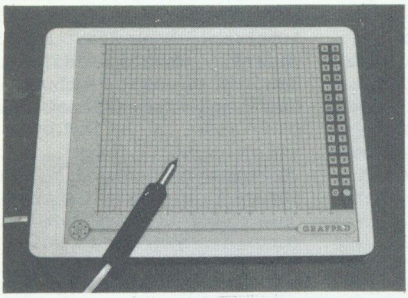
6 3D Text
12 Jigsaw Puzzle
23 Examples on Logic
26 View Printer Driver Generator
31 Disc Back-up Routine
36 Mode 7 Graphics
40 Workshop Procedures
44 EQU Functions for Basic I
46 Hatrace

HINTS, TIPS & INFO

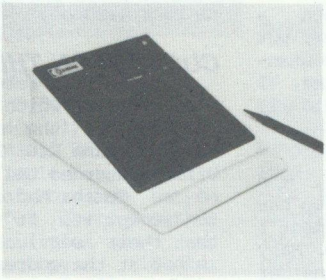
8 Slow Listings
8 Line Number Decoding
15 Validating GET Function
39 Problems with Sidewise
45 Torch Reset
45 Deleting Wordwise Text
45 Interrupting Aries



Mode 7 Graphics



Graphics Tablets



Graphics Tablets



3D Text

EDITORIAL JOTTINGS

One of the ways in which we try to assist members is by negotiating general discounts for BEEBUG members with dealers and retail outlets selling Beeb products. The list of these general dealers' discounts has become too long to publish in every supplement and so, in future, this list will appear in full at two or three monthly intervals with only updates in between.

We also provide special members' offers on selected hardware and software products which we believe to be good value for money – such as Wordwise and other ROMs from Computer Concepts, and more recently on the AMX Mouse. We are now expanding the range of these special offers, and we begin this month with the Symphony Keyboard from ATPL (reviewed in Vol. 3 No. 9) and the Ibico LTR-1 printer (reviewed this month). Details of these latest offers appear in the members' price list with this issue. For Mouse fans, we've added the latest AMX Desk and AMX Utilities to the list. We expect to announce further special offers for members in future issues.

A small apology is necessary this month to Bill Walker whose April spoof 'Backwards Text' we quite inadvertently credited to Bill Wilkinson (a figment of our imagination).

Prize winning hints in this issue earn B.R. Hill £10 and J. Evans £5. We always welcome your hints and tips, so keep them coming.

We shall be at the Acorn User Show (Barbican Centre – 25th to 28th July) with BEEBUG and the full range of BEEBUGSOFT products. We hope to see many of you there.

Finally, we will have moved offices by the time you receive this issue of BEEBUG, still in St Albans but with more space for all of BEEBUG's activities. Our telephone numbers will initially remain the same but there may have been a few hiccups during the move. The postal address remains the same.

PROGRAM CLASSIFICATION

All programs in the magazine, and on the magazine cassette/disc, are marked with the symbols shown below. An uncrossed symbol indicates full working, a single line through a symbol shows partial working (normally some modifications will be required), and a cross through a symbol indicates a program that will not work on that system. We have also a symbol for the new model B+.

Basic I	I	Electron	
Basic II	II	Disc	
Tube		Cassette	
Model B+	B+		

POSTBAG

POSTBAG

MULTI-LINGUAL BEEBUG?

I have been thinking about the BBC micro and languages other than Basic. There are quite a few of these now so I won't list them. I wonder how many members of BEEBUG are also interested in other language options.

My suggestion is that BEEBUG sponsor another magazine which deals specifically with these. With increasing use (which I am sure will come about) of Logo and Pascal, and in the absence of any magazine dealing with these, I am sure that there is, or will be, a need for such a magazine. Please give the idea an airing.

John Maher

Another member, Mr J. D. Muddiman, has also written to us, more specifically with regard to Lisp. We would certainly consider ways in which BEEBUG could better cover other languages if it can be shown that the demand is there.

WHO'S THE APRIL FOOL?

You recently published 'Backwards Text' (Vol.3 No.10) as a "handy little utility to fool your friends". I immediately have not one but two serious applications. As issued it is ideal for the service engineer attending to a monitor, working from the rear and looking into a mirror facing the screen. Obvious ain't it? Further, some applications specifically require a right-to-left operation, e.g.Arabic. Yes, this code is already in development use in this area [a university].

A final plea. In all 'serious work' please do everything possible to ensure programs work on the 6502 second processor.

A.A.Hock

So, even an April Fool program has a serious application! As readers will have seen, we are now testing all magazine programs on the 6502 second processor, and are ensuring that they do work with this system if at all possible.

BACK TO BEEBUG

Having only been a member since October, but impressed with the standard of the magazine, I have recently obtained all of the back issues.

Without a doubt, the small outlay on these has been the best investment I have made on the Beeb. Many of the programs I consider absolutely brilliant, and one or two of these on their own would be worth the cost. Besides these there are all the reviews, and the hints and tips. I can now get LISTO to work, and can get immediate conversions between hex and decimal to name just two.

I trust you will not be too embarrassed to publish this letter, as not only would I like the editorial staff and contributors to know how much their efforts are appreciated, I would recommend that any new member obtains all the back issues.

Programs that I have entered so far that I consider exceptional are: ASTAAD, Demolition, Multiple Character Definer, 3D Rotation, Beebmaze, Sound Wizard, Envelope Editor, Dancing Lines, Patchwork, Mason and Extra GCOL modes.

D.P.Dyer

We normally keep all back issues in stock, but we have recently run out of certain issues of volume one. We are intending to reprint the whole of volume one (see insert for further details of special offers on back issues).

CUMANA ON THE MOVE

I recently returned a disc drive which has a problem to the manufacturer, Cumana Ltd. The Post Office returned the unit to me as Cumana had declined to accept it. It appears that their service centre is not at the address which appears in adverts, and on some guarantee cards, but at Daimler Close, Royal Oak Industrial Est., Daventry, Northants NN11 5QT.

Andrew South

We have checked with Cumana and the address above is now correct.

LOGO FOR SCHOOLS

Schools should note that it is the LSL version of Logo that they may obtain from E.J.Arnold and not the Logotron version as stated in the review of Logo in Vol.3 No.10.

News News News News News News.....

ACORN

The ABC range is to see the light of day again in the form of the Cambridge Workstation range. The new machines, based on the ABC 210 that never actually made it, are aimed at scientific and industrial users. The Workstation is Beeb based with a 32016 second processor, 1Mb RAM, colour monitor, Acorn's Panos OS, Basic, Pascal, C, Lisp, 32016 assembler and a mainframe communications link. The range costs about £2700 with two floppies and moves up to a 10Mb hard disk version. The machines also include a memory management system allowing the easy addition of more RAM and a maths processor chip giving exceptionally high speed. The Workstation range is due for launch soon. Further details might be wrung from Acorn on 0223-245200.

ACORNSOFT

Acornsoft has been very busy recently. A full screen Basic Editor ROM is now available for £30. This offers a Wordwise style editor for Basic programs with facilities such as block copy, labels, renumbering, and so on. Coming soon to your screens are Viewstore (a database for the View family, about £60), Viewspell (a spelling checker for View), Termulator (a terminal emulator, about £35) and the long awaited Graphics Extension ROM (offering sprites, colour mixing, etc for around £30). Beebug has seen preliminary versions of all these and we're happy to say they look promising.

View version 3 is also about to make an appearance. This will be B+ compatible and offer a few advanced features for £90. View 2.1 will continue to be sold. Further details from their new number in Cambridge, 0223-214411.

TANDATA TANDEM

Tandata has increased its range of modems with the BT approved Tm512. Costing £340, the Tm512 offers auto answer, auto dial, auto baud rate (1200 to 9600), and a built-in store of numbers and passwords. In addition, Tandata is offering a free quarter year subscription to Micronet to anyone purchasing any Tandata modem and BBC micropack before 31st July. Further details from Tandata on 06845-68421.

ROM GUARDIAN

Guardian from Pentrix is a ROM that offers you security as well as utilities. A password sign-on system is used to stop unauthorised usage of your Beeb. In addition Guardian offers a range of Basic programming aids (such as verify, compress, relocate), utilities (such as move memory, cursor on/off) and ROM management commands. Guardian costs £30 from Pentrix, who are to be found at PO BOX 65, Solihull, B91 3UP.

DOUBLE DENSITY CP/M

Slogger Software has produced a utilities disk for users of the Z80 second processor and the Opus DDFS. Now your CP/M can handle up to 785K of storage pre disk, with 256 directory entries and with a 25% increase in speed, all for £49.50. Further details from Slogger on 0634-811634.

NEW BOOKS

After 'Wordwise Applications Guide' can only come 'Using Wordwise Plus', also from Norwich Computer Services. All the info on how to use WW+ for a mere £7.00. Contact NCS on 0603-621157

NEW SOFTWARE

Clares' 'Profile' is a procedure management system to create and use a procedure library on disc for use in your own programs. Profile costs £12, but back issues of Beebug showing just how to do it all are much cheaper! The much acclaimed 'Hampstead' from Melbourne House is now available for the Beeb. For £9.95 you can attain the social heights of Hamstead against such adversaries as Justin Perrier and Sir Lionel Thrumm. Rather! Incentive Software has finally released the much hyped 'Confusion' - a rather ordinary sounding arcade game for £6.95. If it's more than ordinary we will be reviewing it here. Another arcade game comes from Alligata in the form of 'Nightworld' for £7.95 and Robico (see 'Banjax' in this issue) has come up with another adventure called 'Assassin' for £9.95. Also on the arcade front is 'Sub Strike' from Tomorrow's Dream for £6.50 and Ultimate has been busy and has produced 'Alien 8', 'Atic Atac' and 'Knight Lore' for the Beeb, all for £9.95. Colonial members will be pleased to hear that US Gold has at last turned its head to the BBC micro with 'Bounty Bob Strikes Back' for £9.95.

Colourful and spectacular three-dimensional titles are readily produced with this program by O. R. Thomas.

Text on the BBC micro's screen is very versatile - you can print just about anything, in lots of colours, and very fast too. However, a serious drawback of the Beeb for the more flamboyant amongst us is the decidedly two dimensional nature of the screen display. This program will give you the next best thing to a holographic TV.

Any text or user defined characters can be displayed on the screen, in mode 1, in a 3-D style with different colours used for the different faces of the letters. The examples shown here will give you the general idea. You can use the 3-D text for title screens, posters, or anything requiring an extra dimension.

The program consists mainly of the procedures that actually plot the 3-D characters (lines 1000 to 1570). All these must be included in your own program. Also listed here is a short demonstration procedure (PROCdemo) to show you how the program is used. Various parameters are set up, a couple of procedures called, and then the main PROC3D is called.

The call to PROCprepare must be made before PROC3D is called. If you want the text to appear centred on the screen then PROCcentre should also be called, as in the demonstration procedure. If the text is to be placed in any other position across the screen then spaces should be added to the END of the text string as it is this end that is normally positioned hard against the right hand side of the screen. Indeed, it is just this action that PROCcentre performs automatically when it centres each line of your text.

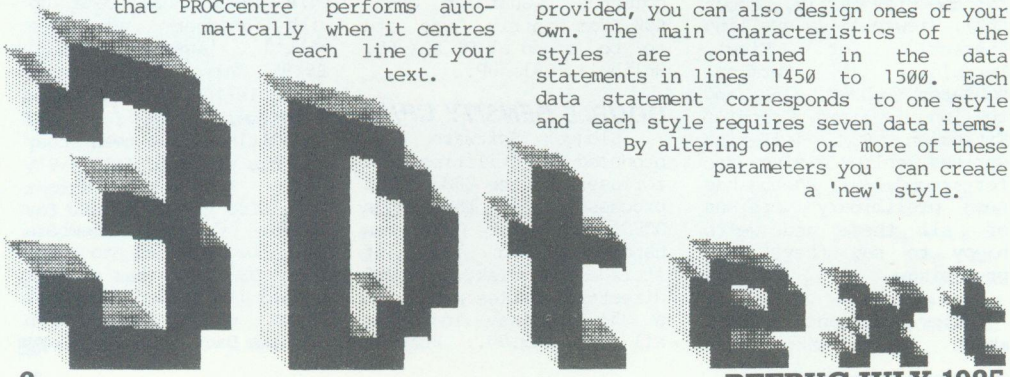
There are several styles of text that the program can produce. These differ in the relative proportions of the letters. Some are double height, some double width and some both. Naturally, the larger the letters, the less will fit onto the screen. You should choose the style suitable for your text and assign the style number to the variable, style%. These are the styles available.

Style	No. of lines on screen	No. of letters per line
1	3	13
2	3	6
3	2	13
4	2	6
5	1	13
6	1	6

The number of possible lines of text actually used to display words should be assigned to the variable, lines% and the text, or characters, required in these lines assigned to elements of the string array, line$, up to the value of line%. The demonstration procedure shows just how all this is done. Now the procedure, PROC3D, can be called to produce the display.

When using the 3-D procedures in your own program you could also make a call to a screen dump routine (such as those in BEEBUGSOFT's Dumpmaster or Computer Concept's Printmaster) after the PROC3D call, if a hard copy of the 3-D text is needed.

As well as the six letter styles provided, you can also design ones of your own. The main characteristics of the styles are contained in the data statements in lines 1450 to 1500. Each data statement corresponds to one style and each style requires seven data items. By altering one or more of these parameters you can create a 'new' style.



The seven data items are read into the seven variables, height, mletters%, mlines%, add, change%, alt%, mult%. These control, respectively, the vertical distance between elements of the characters, the maximum number of letters in a line, the maximum number of lines on the screen, the width of the characters, the height of the elements, the vertical position of the right hand end of the lines, and the distance between lines.

Not all values for all the variables will give exactly what you might expect. For example substituting a value of 2 for height% in the data for style 1 will double the vertical distance between elements, leaving a gap between each. However, using a value of 3 will not produce a smaller gap but the same sized one between elements 3 and 4, and 6 and 7 only!

A bit of experimenting can not only keep you amused for hours but produce some interesting, and even useful results.

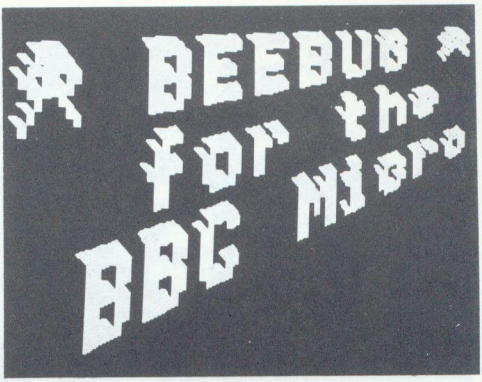
To start you off, try any of the styles with this data:

```
2,7,2,0.5,1,0,20
```

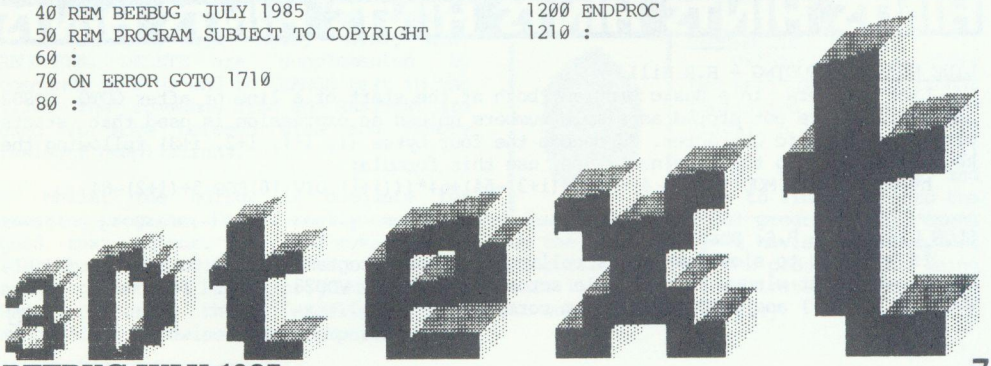
PROGRAM NOTES

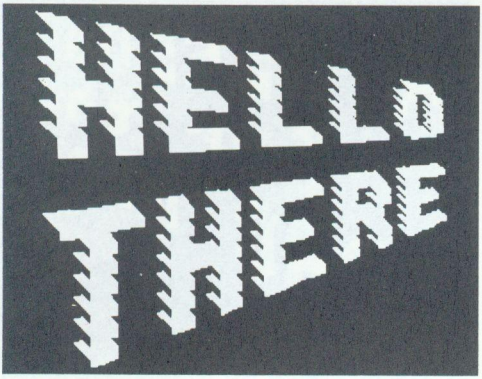
The program determines the shape of the character to plot it onto the screen, not by way of the OSWORD call with the accumulator set equal to 10 as you might expect, but by writing the characters in the normal way at the bottom of the screen and using POINT to see which pixels are lit.

```
10 REM PROGRAM 3D TEXT
20 REM VERSION B0.2
30 REM AUTHOR  O.R.THOMAS
40 REM BEEBUG  JULY 1985
50 REM PROGRAM SUBJECT TO COPYRIGHT
60 :
70 ON ERROR GOTO 1710
80 :
```




```
100 DIMline$(3)
110 MODE 1
120 VDU19,3,2;0;19,2,7;0;
130 PROCdemo
140 END
150 :
1000 DEF PROC3D
1010 FOR L%=lines%-1 TO 0 STEP-1
1020 ang%=alt%-(L%*mult%)
1030 PROCword(line$(L%+1),1008-(32*L%))
1040 NEXT L%
1050 ENDPROC
1060 :
1070 DEF PROCword(line$(L%),oy%)
1080 COLOUR2
1090 PRINTTAB(20,31)line$(L%);
1100 FOR py%=1 TO 33 STEP4
1110 angle%=(py%/height)+ang%
1120 a1=TAN(RAD(angle%)):a2=TAN(RAD(ang
le%-change%))
1130 sub=4:lp=1309:rp=1359
1140 FOR px%=(LEN(line$(L%))+1)*32 TO 0
STEP-4
1150 IF POINT(px%+640,py%)=2 PROCbox
1160 rp=lp:sub=sub+add:lp=lp-sub
1170 NEXT px%
1180 NEXT py%
1190 PRINTTAB(20,31) SPC(18);
1200 ENDPROC
1210 :
```






```
 1220 DEF PROCbox
 1230 PROCface(2,lp,rp,lp,rp,((1579-lp)*
a2)+oy%,((1579-rp)*a2)+oy%,((1579-lp)*al
)+oy%,((1579-rp)*al)+oy%)
 1240 PROCface(1,lp-sub,lp,lp-sub,lp,((1
579-lp)*a2)+oy%+sub,((1579-lp)*a2)+oy%,(
(1579-lp)*al)+oy%+sub,((1579-lp)*al)+oy%
)
 1250 PROCface(3,lp-sub,rp-sub,lp,rp,((1
579-lp)*al)+oy%+sub,((1579-rp)*al)+oy%+s
ub,((1579-lp)*al)+oy%,((1579-rp)*al)+oy%
)
 1260 ENDPROC
 1270 :
 1280 DEF PROCface(col%,x1,x2,x3,x4,y1,y
2,y3,y4)
 1290 GCOL0,col%
 1300 PLOT4,x1,y1
 1310 PLOT4,x2,y2
 1320 PLOT85,x3,y3
 1330 PLOT4,x2,y2
 1340 PLOT4,x4,y4
 1350 PLOT85,x3,y3
 1360 ENDPROC
 1370 :
 1380 DEF PROCprepare
 1390 RESTORE1450
 1400 FOR L%=1 TO style%
 1410 READ height,mletters%,mlines%,add,
change%,alt%,mult%
 1420 NEXT
 1430 IF lines%>mlines% THEN CLS:PRINT "
Too many lines for style ";style%:END
 1440 ENDPROC
 1450 DATA4,13,3,.13,1,-13,8
 1460 DATA4,6,3,.65,1,-13,8
 1470 DATA1.95,13,2,.13,2,-16,15
 1480 DATA1.95,6,2,.65,2,-16,15
 1490 DATA1.33,13,1,.13,3,-29,8
 1500 DATA1.33,6,1,.65,3,-29,8
 1510 :
 1520 DEF PROCcentre
 1530 FOR L%=1 TO lines%
 1540 IF LEN(line$(L%))=mletters% GOTO15
60
 1550 line$(L%)=line$(L%)+STRING$((mlett
ers%+1-LEN(line$(L%)))*.5," ")
 1560 NEXT L%
 1570 ENDPROC
 1580 :
 1590 DEF PROCdemo
 1600 VDU23,230,60,126,219,126,36,66,129
,0
 1610 style%=1
 1620 lines%=3
 1630 line$(1)=CHR$230+" BEEBUG "+CHR$23
0
 1640 line$(2)="for the"
 1650 line$(3)="BBC Micro"
 1660 PROCprepare
 1670 PROCcentre
 1680 PROC3D
 1690 ENDPROC
 1700 :
 1710 ON ERROR OFF
 1720 MODE 7
 1730 IF ERR<>17 THEN REPORT:PRINT " at
line ";ERL
 1740 END
```

HINTS HINTS HINTS HINTS HINTS HINTS

LINE NUMBER DECODING - B.R.Hill

Line numbers in a Basic program, both at the start of a line or after GOTO, GOSUB and RESTORE, are not stored as simple numbers unless an expression is used that starts with a non-numeric character. To decode the four bytes (i, i+1, i+2, i+3) following the keyword token into the line number, n, use this formula:

```
n=4096*((i+1) MOD 16)EOR 4)+256*((i+3)-64)+64*(((i+1) DIV 16)EOR 5+(i+2)-64
```

SLOW LISTINGS - P.S. Bhandari

If you want to slow down the scrolling of a long program listing to a more readable speed set a text window to the whole screen area using VDU28 (e.g. for mode 3 use VDU28,0,31,79,1) and LIST the program normally.

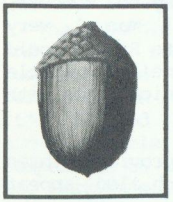
Oxford Pascal

Oxford Computer Systems have produced a challenge to Acornsoft's ISO Pascal for the Beeb. Pascal expert John Maher checks it out.

Oxford Computer Systems Pascal Compiler price £41.95 (cassette), £61.95 (disc), including VAT, p & p.

The BBC micro now has four versions of Pascal, together with at least two others which will run on second processors only (TDI's UCSD P-System Pascal and Borland's Turbo Pascal). However, of the four alternatives for first processor operation neither Acornsoft's restricted version, called S-Pascal, nor the very non-standard Pascal-T ('Tiny Pascal') can be recommended, so that for most people the choice will be between 'ISO' Pascal from Acornsoft and OCSS's 'Oxford' Pascal.

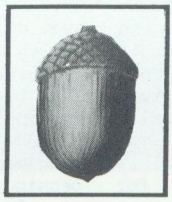
There are two versions of Oxford Pascal, one for cassette operation, the other for use with discs, and it does operate on the 6502 second processor, though it will not run on the Electron yet. Supplied with the Oxford system are a 50 page manual, a 16K ROM and a cassette or 40 (or 80) track disc. The system is claimed to work with both Basic I and II, and with all known DFSs.



THE EDITOR.

Pascal text files are entered in Oxford Pascal with a simple line editor, which is a slightly extended version of BBC Basic's editor. Programs are given line numbers, then AUTO, LIST, NEW, RENUMBER, DELETE are supplemented by commands to FIND and to CHANGE text in the file. The commands DISK and RESIDENT are used to move between disc and memory resident compilations.

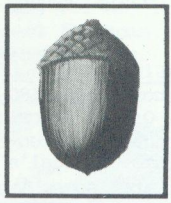
Whilst the editor is adequate for entering programs, Pascal really needs a good text editor. I found myself using the ISO Pascal editor, or Wordwise for entering Oxford Pascal programs. The ISO Pascal editor is really excellent, very much like Wordwise. By comparison the Oxford Pascal editor is very limited. However, Oxford Pascal only occupies one sideways ROM slot, whereas ISO Pascal occupies two, one of which includes the editor. Otherwise Oxford Pascal has a nice line editor which is quite adequate for simple programs.



COMPILATION

As with ISO Pascal, compilation is to an intermediate code known as p-code. The ROM contains an interpreter for the p-codes (occupying 10K of ROM), this interprets the p-code as the program runs. P-code is very efficient so far as space is concerned, faster than Basic, but not as fast as machine code.

Two compilers are supplied for the disc version of Oxford Pascal, one is memory resident, leaving just over 13K space for the source program and object (p-code) files, while the other is disc based so that 23K is left (both compilers use mode 7). The cassette version only works with the resident compiler, which of course restricts program size drastically. Both Oxford and ISO Pascal allow you to chain in large source files during compilation from disc. However, Oxford Pascal also makes it possible to build larger programs since (similar to BBC Basic) program chaining of code files is possible. Chaining is an important and very useful facility, since it enables segmentation of large programs. During chaining the Pascal global variables are maintained so that programs can communicate with each other. The only trick required is to set up identical constant and variable declarations in the programs.



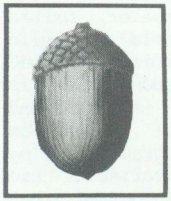
LINKING AND LOADING

Oxford Pascal provides two very useful additional facilities over ISO Pascal. A 'linker' enables you to compile several Pascal source files, and then to link them into the final program - this opens up the possibility of having a library of code files. Whilst the procedure for doing this is a little complicated, I found that it worked satisfactorily.

The 'locate' command enables the programmer to prepare a disc file which can be run independently of Oxford Pascal's ROM, that is 'locate' provides a run time system for the program. Since OCSS make no copyright claims on such programs, this is a very useful facility.

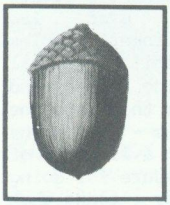
'Locate' has a second function whereby code from a compiled program can be set up to work from a chosen position in the memory. The ROM is still needed for this form of operation, and programs can be loaded and run from the Basic operating system, so that 'mixed' Pascal and BBC Basic (and assembly language) is possible, and there is a facility for passing parameters between such programs. In assembly language, parameters are passed on the Pascal stack.

My overall impression of the Oxford Pascal compiler was that, whilst it is slightly faster than that for ISO Pascal, it is not quite as good. It does not have the range of options available with ISO Pascal, its error messages are weaker, and because it does not stop at each error message a lot of consequential errors appear. I found myself reverting to the ISO Pascal compiler in debugging programs.



BENCHMARKS

On average Oxford Pascal carries out the PCW Pascal benchmarks about 7% slower than ISO Pascal. However, there are some curious individual variations. WHILE and REPEATs are much faster in ISO Pascal, but the arithmetic, maths and algebraic operations are faster in Oxford Pascal. Note that MAXINT in Oxford Pascal is 32767, whereas in ISO Pascal it is 2147483647. The different integer sizes will affect the benchmarks. Reals occupy five bytes in both Pascals.



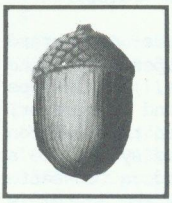
PASCAL EXTENSIONS.

Both Acornsoft ISO Pascal and Oxford Pascal provide language extensions. Oxford Pascal's extensions are somewhat more numerous than those for ISO Pascal, comprising hexadecimal constants, hexadecimal input and output, peek and poke, bit manipulation of integers with 'and', 'or', 'xor', 'shl', 'shr'.

Another interesting form of memory access, an 'origin' procedure, to deal with setting a pointer to a particular memory location, is not explained very well in the manual but looks useful. For instance it should be possible with the aid of this to transfer picture files between disc and screen very rapidly.

Oxford Pascal provides pre-declared procedures for the sound, envelope, move, draw, plot, mode, and gcol Basic commands. A pseudo-random number generator is provided, giving numbers between 0 and 255.

One major region in which Oxford Pascal does gain over ISO Pascal is in string input. Both Pascals require that strings are set up as 'packed array[1..n] of char', but whereas a whole string can be read in Oxford Pascal, a character by character input is needed for ISO Pascal.



THE USER MANUAL

This is a slim paper bound book. It is adequate, but not very well indexed or arranged. It also contains a number of textual and factual errors or omissions. In trying to provide a beginners guide to Pascal, and a description of Oxford Pascal and the operating system and editor, it tries to do too much in too short a space. However, I do not rate the ISO Pascal manual very highly either! Newcomers to Pascal could benefit from P.J.Brown's excellent little book 'Pascal from Basic', which comes with the ISO Pascal system.

The program manual provides three simple program examples which also appear on the disc. Some more examples are desirable, in particular to provide more information on program chaining, location and linking. An example of how to use assembly language via the Basic inline assembler would also be very useful.

OVERALL IMPRESSION

Oxford Pascal lives up to quite a lot of the 'hype' in its adverts, and now that many of the bugs in early versions have been eliminated has much to recommend it.

REVS

REVS is the new super game from Acornsoft, to follow the highly successful and acclaimed ELITE. Mike Williams, keen to try his hand, reports on his experience of driving a Formula Three racing car round the tricky Silverstone circuit.

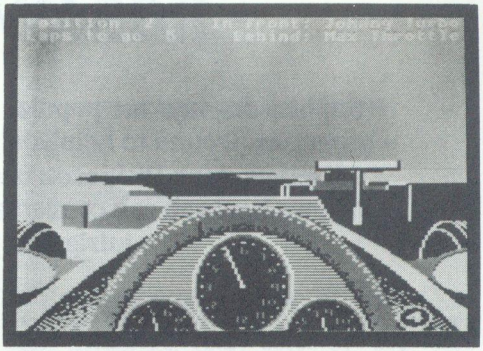
Title : REVS
Supplier : Acornsoft
Price : £14.95 (cassette), £17.65 (disc)
Rating : *****

When Acornsoft offered me the chance to drive a modern Formula 3 racing car around the taxing Silverstone circuit I jumped at the chance. As you've guessed, it wasn't quite the real thing, but Acornsoft's latest super-game REVS.

This must be one of the most realistic simulations of a racing car yet devised for a home micro. The screen displays the view you would get sitting in the cockpit of the racing car, with steering wheel and rev-counter immediately in front of you, and the track curving away into the distance. Two keys control the movement of the steering wheel, left or right, two more keys move up or down the gears (5 gears plus reverse), and two more keys provide accelerator and brakes (a joystick can be used instead).

The controls took some getting use to, and I had completed, or attempted to complete, a good many practice laps before I even found fourth and fifth. It's a good job that practice sessions are available. Racing cars are very responsive to the controls and it is only too easy to spin the car off the track into the perimeter fence. Your lap time during practice is recorded and your best lap time displayed on the screen.

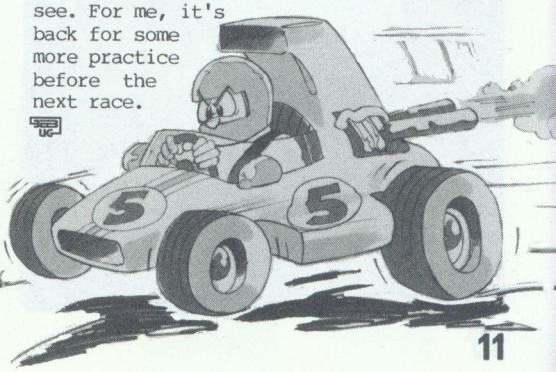
The track itself is an accurate representation of the Silverstone circuit and is complete with all the marker signs indicating approaching bends etc. A nice touch here is that if you spin and end up going the wrong way round the circuit, the signs are now all blank, since you're looking at the reverse side, but it does let you know which way you're going!



Once you feel you have mastered the car and familiarised yourself with the course, you can enter the Formula 3 World Championships against such inspiring drivers as Hugh Jengine and Miles Behind. There are three classes of race, from novice to professional, and you can select the duration of the race in laps.

A final practice session is allowed, which determines your position on the starting grid and then the race is away. From then on only your driving skill counts. Nineteen other cars take part and it can be most disconcerting to see faster cars approaching from behind in the wing mirrors and then zooming past and into the distance. With my lack of racing experience I often found it easier to follow another car round a bend to achieve the correct line.

Alas, in my first race I finished last of the 20 entrants, and that's despite taking many 'short cuts' across the corners. This is undoubtedly a classic game from Acornsoft, to follow Aviator and Elite, though whether it will capture the public's imagination we must wait and see. For me, it's back for some more practice before the next race.



JIGSAW PUZZLE

Graphics displays are popular with most micro users, and many programs abound to help you generate exciting and fascinating displays. Alan Dickinson has come up with a new twist and shows how you can convert any of your best displays into an intriguing jigsaw puzzle.

Jigsaw is an attractive program that jumbles any mode 2 screen picture and then lets you re-build the picture. This program acts as an electronic jigsaw, with the added bonus of actually having all of the bits displayed on the screen (no searching for the last bit that's been vacuumed up!)

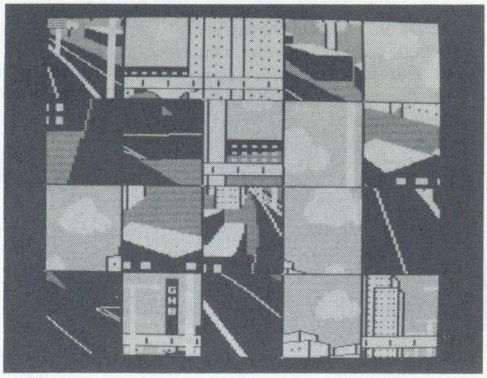
The program is very simple to use, and once you have any mode 2 screen saved, you can experiment with this. The program starts off by asking whether you wish to attempt an easy or hard jigsaw. This relates to the number and size of the pieces that the jigsaw is to contain. After entering this, the program asks for the picture's filename, so that it can load this from cassette or disc. On cassette, the loading of the picture will take quite a long time - so be patient.

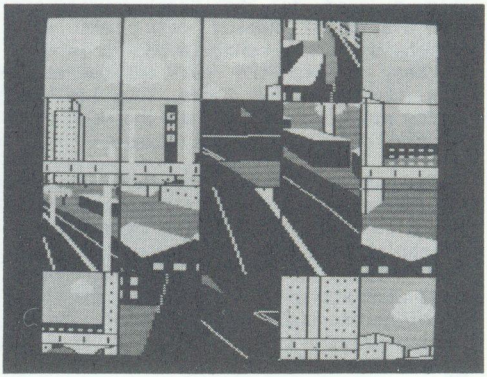
Once the picture has been loaded, the computer waits (while you carefully study the screen), for you to press the space bar to jumble the picture.

To move the cursor (signified by a box of horizontal bars) around the picture use the four cursor keys. When you have positioned the cursor over the piece that you require, just press the space bar. The cursor will now change to a row of vertical bars, and you must now select the position in which you wish to place your jigsaw piece (again using the four cursor keys). Once this has been done, press the space bar again, and the pieces will be automatically swapped.

The program will announce audibly when you have successfully completed the jigsaw (and this is not as easy as it sounds). Press the space bar to return to the start of the program so that you can try another puzzle. One small point to note is that the final picture must correspond exactly with the original. If you have two identical looking pieces (for example sky), then these pieces must be positioned correctly, even if there is no difference visually.

If you have trouble in getting exactly the original picture, pressing Escape will take you back to the start of the program.





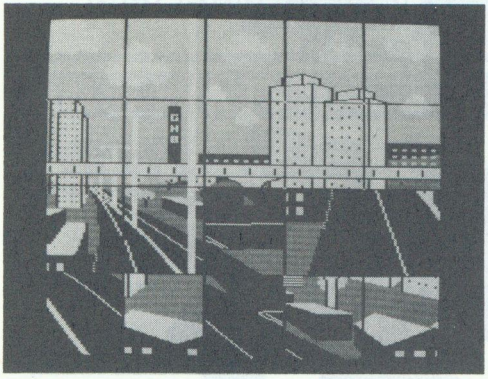
To get you going we have included a picture file on this month's magazine cassette/disc so that you can try out the program straight away.

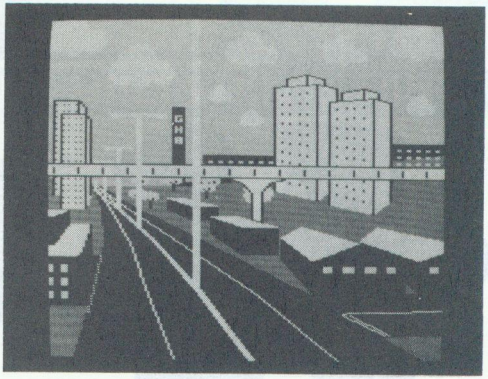
PROGRAM NOTES

For the more technically minded here is a list of some of the procedures and their uses.

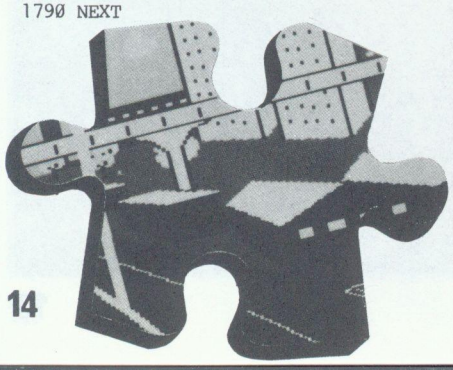
options	Display the initial page and select easy or hard game.
easy	Set the computer up for an easy game.
hard	Set the computer up for a hard game.
jigsaw	Load the required picture into memory.
init	Set up the jigsaw.
grid	Place a grid on the jigsaw.
jumble	Mix the jigsaw up into the required number of pieces and randomly position them.
sort	Test if jigsaw complete.
swap	Swap two pieces of the jigsaw around.

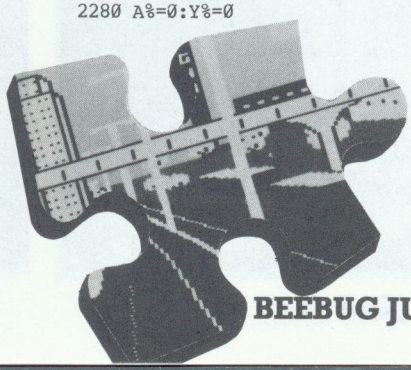
```
   10 REM PROGRAM JIGSAW
   20 REM VERSION B0.3
   30 REM AUTHOR  Alan Dickinson
   40 REM BEEBUG July 1985
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 T$=" J I G S A W   P U Z Z L E S"
  110 esw%=0
  120 PROCfile:PROCassm
  130 DIM T%(79)
  140 DIM param 31,c1 31,c2 31
  150 ON ERROR IF ERR<>17 AND ERR<>195 A
ND ERR<>200 THEN 210
  160 REPEAT
  170 MODE7:PROCoptions
  180 MODE2:PROCjigsaw
  190 UNTIL 0
  200 :
  210 MODE 7:REPORT:PRINT" at ";ERL
  220 *FX4
  230 END
  240 :
 1000 DEFPROCquit
 1010 VDU26
 1020 CLS
 1030 END
 1040 :
 1050 DEFPROCoptions
 1060 *FX4
 1070 VDU129,157,10,13
 1080 VDU131,157,132,141:PRINT T$
 1090 VDU131,157,132,141:PRINT T$
 1100 VDU129,157,10,13
 1110 VDU129,157,13,10
 1120 IF esw%=1 REPORT ELSE esw%=1
 1130 VDU28,2,24,37,12
 1140 PRINTCHR$133;"Select Jigsaw Diffic
ulty"''';TAB(7)"Easy"''TAB(7)"Hard"''TAB(
4)"or Quit"''TAB(7);
 1150 REPEAT
 1160 D$=GET$:SOUND&11,-10,100,1
 1170 PRINT D$;
 1180 UNTIL D$="E" OR D$="H" OR D$="Q"
 1190 IF D$="Q" PROCquit
 1200 IF D$="E" PROCeasy
 1210 IF D$="H" PROChard
 1220 CLS
 1230 REPEAT
 1240 REPEAT
 1250 PRINT'TAB(7)"Picture name ";
 1260 INPUT""F$:SOUND&11,-10,100,1
 1270 UNTIL F$<>""
 1280 IF LEFT$(F$,1)="*" PROCoscli(F$)
 1290 UNTIL LEFT$(F$,1)<>"*"
 1300 ENDPROC
```



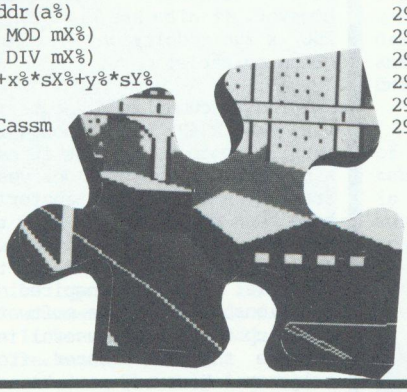


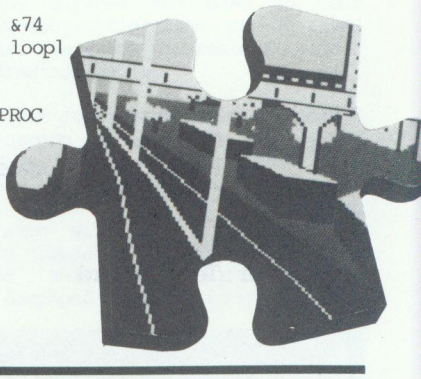
```
1310 :
1320 DEFPROCeasy
1330 nX%=128:nY%=8:d%=256
1340 mX%=5:mY%=4:mN%=20:mJ%=40
1350 sX%=128:sY%=5120
1360 ENDPROC
1370 :
1380 DEFPROChard
1390 nX%=64:nY%=4:d%=128
1400 mX%=10:mY%=8:mN%=80:mJ%=100
1410 sX%=64:sY%=2560
1420 ENDPROC
1430 :
1440 DEFPROCoscli(p$)
1450 $param=p$
1460 X%=param MOD 256
1470 Y%=param DIV 256
1480 CALL &FFF7
1490 ENDPROC
1500 :
1510 DEFPROCjigsaw
1520 VDU23,1,0;0;0;0;
1530 FORJ%=8 TO 15
1540 VDU19,J%,0,0,0,0
1550 NEXT:*OPT 1,0
1560 *FX4,1
1570 A$="*LOAD "+F$+" 3000"
1580 PROCoscli(A$)
1590 GCOL2,135:CLG
1600 A$=GET$
1610 PROCinit
1620 PROCgrid
1630 PROCjumble
1640 PROCsort
1650 PROCgrid
1660 FOR J%=1 TO 255
1670 SOUND&11,-10,J%,1
1680 SOUND&12,-8,-J%,1
1690 SOUND&13,-8,J%+12,2
1700 NEXT
1710 A$=GET$
1720 esw%=0
1730 ENDPROC
1740 :
1750 DEFPROCinit
1760 FOR J%=0 TO 31 STEP4
1770 c1!J%=&003F003F
1780 c2!J%=&15151515
1790 NEXT
1800 FOR n%=0 TO mN%-1
1810 T%(n%)=n%
1820 NEXT
1830 ENDPROC
1840 :
1850 DEFPROCgrid
1860 GCOL3,8
1870 FOR j%=0 TO 1279 STEP d%
1880 MOVE j%,0:PLOT1,0,1023
1890 MOVE 0,j%:PLOT1,1279,0
1900 NEXT
1910 ENDPROC
1920 :
1930 DEFPROCjumble
1940 FOR j%=1 TO mJ%
1950 a%=RND(mN%)-1
1960 REPEAT
1970 b%=RND(mN%)-1
1980 UNTIL b%<>a%
1990 PROCswap(a%,b%)
2000 NEXT
2010 ENDPROC
2020 :
2030 DEFPROCsort
2040 X%=0:Y%=0
2050 REPEAT
2060 a%=FNpos(c1):SOUND&11,-8,100,1
2070 b%=FNpos(c2):SOUND&11,-8,200,1
2080 PROCswap(a%,b%)
2090 FOR j%=0 TO mN%-1
2100 IF T%(j%)<>j% j%=99
2110 NEXT
2120 UNTIL j%=mN%
2130 ENDPROC
2140 :
2150 DEFFNpos(curs%)
2160 REPEAT:*FX21
2170 PROCcursor(curs%)
2180 K%=GET
2190 PROCcursor(curs%)
2200 IFK%=139 AND Y%>0 Y%=Y%-1
2210 IFK%=138 AND Y%<mY%-1 Y%=Y%+1
2220 IFK%=136 AND X%>0 X%=X%-1
2230 IFK%=137 AND X%<mX%-1 X%=X%+1
2240 UNTILK%=32
2250 =X%+Y%*mX%
2260 :
2270 DEFPROCfile
2280 A%=0:Y%=0
```





```
 2290 A%=(USR&FFDA) AND &FF0000 DIV &100
00
 2300 IFA%<3 code=&B00 ELSE code=&900
 2310 ENDPROC
 2320 =X%+Y%*mX%
 2330 :
 2340 DEFPROCcursor(curs%)
 2350 P%=FNaddr(X%+Y%*mX%)
 2360 ?&70=P% MOD 256:?&71=P% DIV 256
 2370 ?&72=curs% MOD 256
 2380 ?&73=curs% DIV 256
 2390 ?&74=1:?&75=32:CALLcode
 2400 ENDPROC
 2410 :
 2420 DEFPROCswap(a%,b%)
 2430 P%=FNaddr(a%)
 2440 Q%=FNaddr(b%)
 2450 ?&70=P% MOD 256
 2460 ?&71=P% DIV 256
 2470 ?&72=Q% MOD 256
 2480 ?&73=Q% DIV 256
 2490 ?&74=nY%
 2500 ?&75=nX%
 2510 CALLcode
 2520 Z%=T%(a%)
 2530 T%(a%)=T%(b%)
 2540 T%(b%)=Z%
 2550 ENDPROC
 2560 :
 2570 DEFFNaddr(a%)
 2580 x%=(a% MOD mX%)
 2590 y%=(a% DIV mX%)
 2600 =HIMEM+x%*sX%+y%*sY%
 2610 :
 2620 DEFPROCassm
 2630 P%=code
 2640 [OPT 2
 2650 LDX #0
 2660 .loop1
 2670 LDY #0
 2680 .loop2
 2690 LDA (&70),Y
 2700 STA &76
 2710 LDA (&72),Y
 2720 STA (&70),Y
 2730 LDA &76
 2740 STA (&72),Y
 2750 INY
 2760 CPY &75
 2770 BNE loop2
 2780 CLC
 2790 LDA #(640 MOD 256)
 2800 ADC &70
 2810 STA &70
 2820 LDA #(640 DIV 256)
 2830 ADC &71
 2840 STA &71
 2850 CLC
 2860 LDA #(640 MOD 256)
 2870 ADC &72
 2880 STA &72
 2890 LDA #(640 DIV 256)
 2900 ADC &73
 2910 STA &73
 2920 INX
 2930 CPX &74
 2940 BNE loop1
 2950 RTS
 2960 ]
 2970 ENDPROC
```





HINTS HINTS HINTS HINTS HINTS HINTS HINTS

VALIDATING GET FUNCTION - David Reed

This is a short routine to check a key-press for one of two letters in either upper or lower case.

```
  100 DEF FNkey(K$)
  110 LOCAL K%
  120 REPEAT
  130 K%=INSTR("@"+K$,GET$) DIV 2
  140 UNTIL K%
  150 =K%
```

For example: FNkey("YyNn") will return a 1 for 'Y' or 'y', a 2 for 'N' or 'n' and will ignore any other key-press.

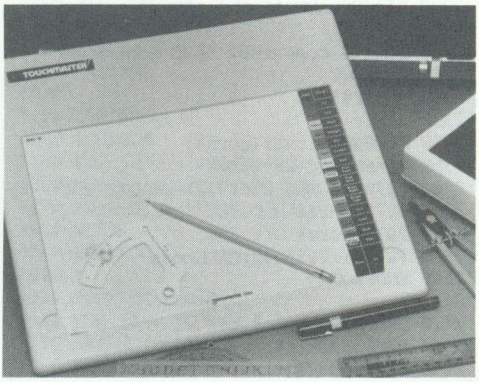
Graphics Tablets

We report on three graphics input devices, the well established Graphpad, and the more recent Touchmaster and Cumana Touchpad.

Product : Touchmaster Pad
Supplier : Touchmaster
P.O. Box 3, Port Talbot,
West Glamorgan, SA13 1WH.
0656-744770
Price : £149.95
Reviewer : Geoff Bains

Product : Touchpad
Supplier : Cumana
Pines Trading Estate,
Guildford, GU3 3BH.
0483-503121
Price : £69.95
Reviewer : Terry Hallard

Product : Graphpad
Supplier : British Micro
Penfold Works, Imperial Way,
Watford, WD2 4YY.
0923-48222
Price : £143.75
Reviewer : C. Chan



TOUCHMASTER

The Touchmaster tablet is marketed more as an alternative keyboard than a graphics tablet. However, graphics tablet it certainly is. The tablet area is about A4 sized, housed in a good quality plastic case and is 'drawn' on with the plastic stylus provided, a pen, or even a finger. It can be used as a child's substitute for a keyboard with the various pieces of software and overlays that Touchmaster sell (Othello, Speli-copter, etc.), however, it also has a very respectable 256 x 256 resolution and can operate as a graphics tablet.

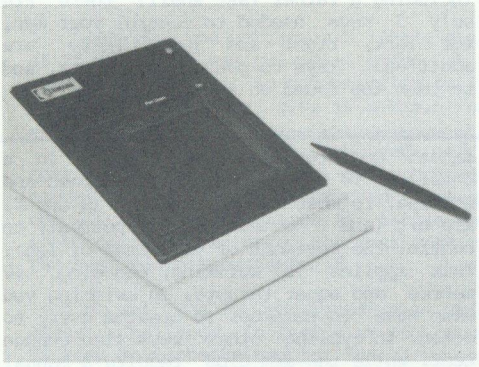
The Touchmaster connects to the Beeb via the RS423 port, though it also has a parallel output. In fact it can connect to a number of micros. It derives its power from a small plug/transformer. It would have been much nicer to have the Beeb as the power source. A drawing program (Multipaint) is included in the package. This is not very inspired. The hardware functions well but the software does not live up to it. More useful is the ability to use the Touchmaster from your own programs. However, next to no information is given about this in the manual. If Touchmaster cannot produce good software and no one else is interested, then there is no excuse for not giving more information for a DIY job.

As a large, child's keyboard the Touchmaster does very well, but at £150, it outprices itself from this market. As a graphics tablet it can also function well, but until more software to constructively make use of this fact arrives, it will remain a white elephant. Now if AMS was obsessed with tablets rather than mice...

TOUCH PAD

Cumana's Touch Pad is much smaller than Touchmaster (about 2.5 inches square) on a wedge case. It is nicely finished in cream and black and comes with a handbook, a cassette and leads to connect into the RS423 socket for input and, for power, into the auxiliary power socket. An adaptor is thoughtfully supplied to allow a disc drive to connect to this as well.

On the top there are two LEDs ('Power On' and 'Pen Down') and two rectangular buttons with arrows on, though only one is used. In the centre of the sloping top is the tiny tablet area, and a black 'inert' plastic stylus completes the setup. Surprisingly the stylus does not mark the plastic of the tablet even though considerable pressure is required to keep the Touch Pad transmitting - lift the stylus and drawing stops. Because of the size of the pad, tracing is impractical.



A graphics program is included, offering all commands selected from a touch menu. It's not very good. It's limited to freehand, straight lines, squares and circles. The only innovation is RAYS which draws radial lines from a centre to a sequence of stylus positions.

The manual gives the usual 'how' instructions and notes for programmers. What it does not say is 'why'. There is no use for this device that is not covered better by other tablets. It is not very sensitive in its screen positioning because of the very small drawing area and the constant pressure needed. In use, its action on the screen is similar to a very stiff joystick. While it looks very nice, for its size it is expensive.

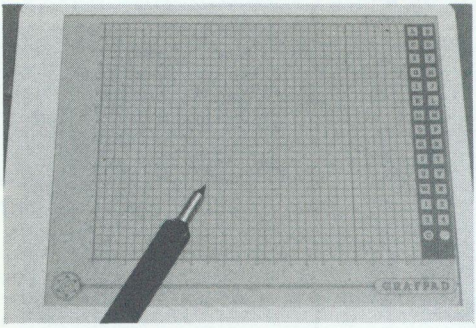
GRAPHPAD

Graphpad is an A4 sized tablet which connects to the Beeb via the user port. The tablet surface is protected by a perspex sheet under which diagrams can be inserted for tracing. An 'electric' pen is used. Pressing the pen onto the surface makes a contact detectable by the micro. This tends to scratch the perspex, but glass could be used instead.

The surface markings are divided into a grid area and a touch pad area for menu selections. The whole tablet has a resolution of 320 x 256.

With the tablet come three programs on cassette or disc. The first program is an introductory one that displays the pen's co-ordinates, places a cursor on the screen and says whether the pen is up or down. The second and third programs are drawing programs. You can create basic shapes, draw freehand, change colours and dump the results to a printer. Only mode 1 is used and, although there is a touch pad on the tablet itself, you still need to use the keyboard to select options like colours, shapes, etc. Large characters (16 x 16) can be defined and used in your drawing. These definitions can be saved and several sets of characters loaded during a drawing session.

The manual consists mainly of listings of the programs and a routine to read the tablet, to use in your own programs. This is the only software with any lasting use but it is difficult to write effective programs even with this routine. Until third party companies can be persuaded to write for this excellent piece of hardware, Graphpad will remain underused.



Title : Blitzkrieg
Supplier: Software Invasion
Price : £7.95 (cass.)
£9.95 (disc)
Reviewer: Geoff Bains
Rating : ***

Battlezone was an arcade classic in its time but the wire frame graphics it boasted look less inspiring today. Blitzkrieg is an attempt to bring the game up to date. Gone are the wire frames and in their place are graphics created with models and a video digitising system.

The effect is quite good, if only monochrome (in green), and both the enemy tanks and your machine look impressive. The scrolling scenery in the background is also very detailed and nicely shaded but is totally inaccessible. You cannot move your tank, only rotate the turret. The enemy tanks rumble around you, though, and are surprisingly slow at getting you within their sights. Fortunately they will

Title : Contraption
Supplier : Icon Software
Price : £8.95
Reviewer : Alan Webster
Rating : ****

Contraption is another cartoon style arcade game from Icon, the publishers of 'Caveman Capers'. The game is similar in concept to 'Manic Miner' and a thousand other similar games. You have to guide your little man, a mad, balding scientist in this case, around a room collecting all the apples before leaving via a door to the next room.

The graphics are outstanding, and the game is very challenging. There are 11 different screens and a host of nasties including a rather fast snail! There are only 3 keys needed to control your man, for left, right and jump. There are additional keys to pause and restart, and to turn the sound on and off.

Title: Brian Jacks Superstars Challenge
Supplier : Martech
Price : £7.95
Reviewer : Mike Williams
Rating : ***

Well, I've just managed to beat Brian Jacks in the Superstars Challenge! How did I manage it in my far from fit state? It was all courtesy of Martech's latest computer game. You compete in a total of 10 events of skill, speed and stamina. In most events two keys (Z and X) are hit (and I do mean hit) repeatedly to control the movement of your arms or legs. This applies in swimming, canoeing, 100 metres and squat thrusts. In swimming you also have to remember to breathe every so often. In cycling other keys also change gear, while the arm dips require a manual dexterity that has to be felt to be believed. The archery at least gives you time to get your breath back.

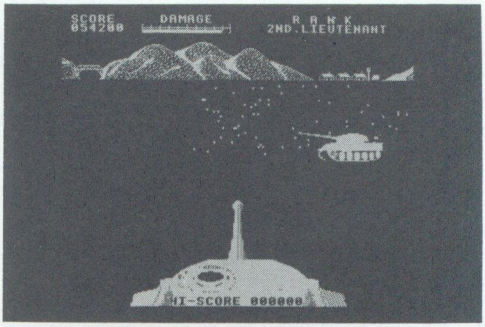
Title : Magic mushrooms
Supplier: Acornsoft
Price : £12.95 (cass.)
£14.95 (disc)
Reviewer: Geoff Bains
Rating : ****

Guide Murphy around the screen of ladders, platforms, moving floors, trampolines, and so on, to collect all the mushrooms. Doesn't sound too original, and certainly not worth ****. However, as you'd expect from Acornsoft there is more to Magic Mushrooms than just this. When you have mastered the nine screens of the platforms and ladders game supplied, you can edit the screens or design entirely new ones of your own.

The game itself is good, and not that easy to master. The graphics are bright and smooth as we have come to expect from Acornsoft. However this tired concept for a computer game is given a whole new lease of life by the screen designer. New

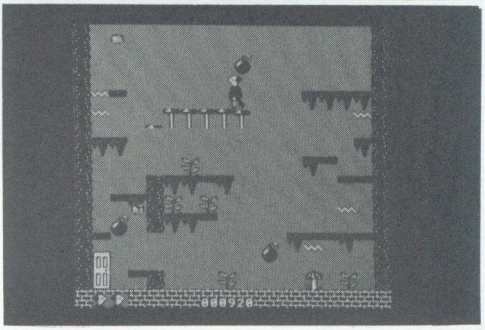
only shoot at your tank when you can see them. You can suffer ten hits from the enemy before your machine finally explodes.

The game is well implemented with a good 3D moving shell effect and excellent graphics. However, Battlezone was around an awful long time ago and the idea is wearing a little thin now. If you like this kind of game then Blitzkrieg will appeal, otherwise you'd be better off lining up something else in your sights.



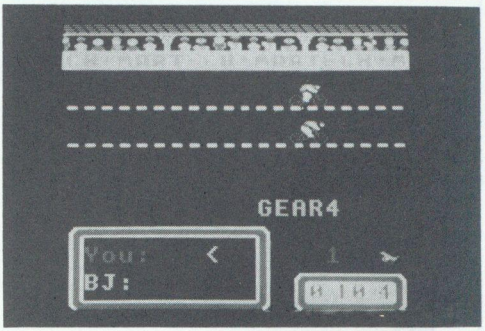
The game is similar in concept to FRAK! and plays as well, if not better. The colour combinations haven't been used to the best effect with an excess of red, yellow and green. The collision detection is as infuriating as it was in FRAK! It can be very annoying when you suddenly die, and the object that killed you missed by a mile.

Overall, the price is a little steep at £8.95. If it was priced at under £8 I wouldn't hesitate in recommending it. If you've got the money, give it a try.



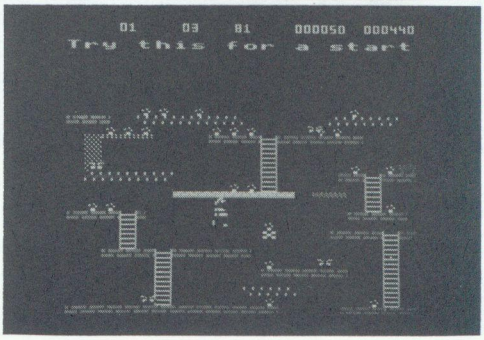
The events are well displayed, and in most a Brian Jacks look-alike competes alongside you. The documentation is meagre but adequate, though there is little good coaching to be had here. You must work out the best techniques yourself.

I'm not really convinced that Brian Jacks' Challenge is really strong enough to keep me entertained for that long, but simulations of sporting events are quite popular and this is one of the better ones. I should know. By now I must have the fittest fingers around!



screens can be created from old ones or from nothing. The screen editor is very simple to use. You just move the cursor around the screen depositing any of the sixteen different symbols as you go. The mushrooms are placed randomly by the Beeb for each game. You can test out a screen from within the editor, and when satisfied, save it and incorporate it in the game itself.

Many games are getting a bit vapid these days. Magic mushrooms has a novel approach and is excellently implemented.



Title : Wizadore
Supplier : Imagine
Price : £7.95
Reviewer : Alan Webster
Rating : ****

Wizadore is another in a long line of recent 'arcade adventures'. This one is from Imagine, a new name in BBC software but well established for other micros. Wizadore is a serious competitor to Micro Power's 'Castle Quest'. The graphics are good and well thought out and the game is as much a challenge as 'Castle Quest'.

The object of Wizadore is to manoeuvre a wizard about the screen, collecting 3 scrolls and a sword, and to destroy the evil dragon Smaun. The only drawbacks with this game, and the reason I gave it 4 stars instead of 5, are the poor key layout, and the difficulty in picking up objects. To move your wizard left and right you must use the corresponding

---

Title : Banjax
Supplier: Robico Software
Price : £9.95 (cass.)
£11.95 (disc)
Reviewer: Mike Williams
Rating : ***

Banjax is a rather nifty bear whose quest is to collect treasures and ultimately find his way to the inner sanctum of the Golden Temple. This game, an example of the current vogue for combining arcade and adventure game elements together, has some of the most superlative graphics I have seen for some time. Regrettably, the game itself does not really live up to its initial expectations.

Banjax the bear starts in front of the castle entrance. From here you can then explore some 240 different locations, collecting all the treasures you can find to build up your score. The screen does not scroll. Instead, each scene is cleared and replaced by a new one, complete with a

---

Title : Sorcery
Supplier : Pace Software
Price : £8.54
Reviewer : Alan Webster
Rating : ***

In Sorcery, you, the noble knight, must rescue a 'fair maiden' from near certain death by the sorceror whilst avoiding the deadly fire-breathing dragons, ghosts, arrows and other nasties. The 'fair maiden' referred to on the cassette label is rather grotesque and definitely not worth risking your life for.

Sorcery is a one player game very much in the mould of Micro Power's 'Killer Gorilla'. Your quest starts at the bottom of the screen, and you must make your way to the top of the screen via sets of steps and walkways. On your way you must collect the treasure that is 'floating' about. Once all the treasure has been collected, you can then finish the screen and go onto the next one.

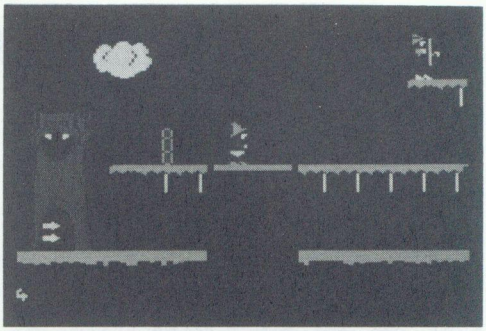
Title : Labyrinth
Supplier: Acornsoft
Price : £12.95 (cass.)
£14.95 (disc)
Reviewer: Geoff Bains
Rating : ****

Maze games are old hat. Labyrinth, however, combines elements of pure maze traversing with touches of adventure and more than a smattering of arcade action. The game is set in a brightly patterned maze (one room on the screen at once) inhabited by various vicious beasts, items of food, and the odd gem. Armed only with a block that can be pushed around in front of the horribly cute character, you must find the gem to get past the force field into the next stage of the maze (where another gem and more tasties and nasties are waiting).

You have limited energy which must be continually replenished with the food and a gun to shoot the baddies, though this also uses up a lot of energy - crushing

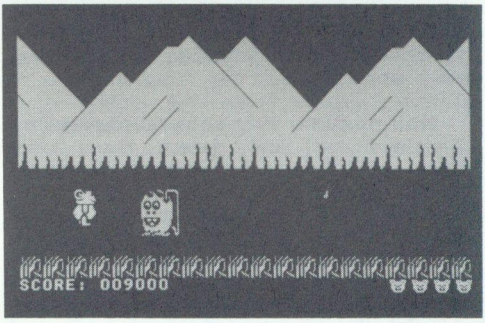
cursor keys. 'A' and 'Z' move up and down with 'Shift' for fire and 'Delete' to pick up objects. For those of us who aren't left-handed this can be very tricky to master, and even after playing it for some time now I still cannot control the wizard properly.

Imagine are running a competition, like Micro Power, in which £100 can be won each month up until the end of August. Overall, the game was fun to play but frustrating to master. If you can cope with the awkward keys then buy it!



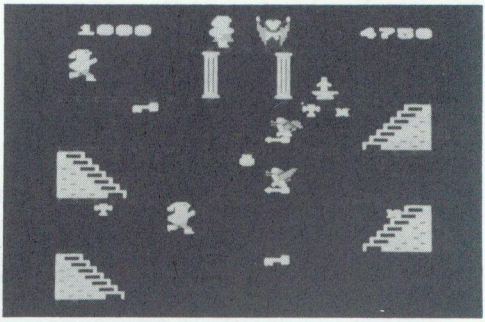
variable number of hazards, some just obstacles, others more lethal. However, repeatedly switching between adjacent scenes will eventually provide a hazard free environment, and you can progress to the next location. This is a rather weak feature of the game, as any challenge can be avoided, rather than solved.

The game is well packaged, complete with an adventurer's notebook, and without doubt the graphics are a real delight, but the game - well for me once I had found the inner sanctum the challenge had gone.



If you are hit by one of the 'nasties', you turn into a frog. To revert back to your normal self, you must guide the maiden from the top of the screen down to the bottom so that she can kiss you to release the spell.

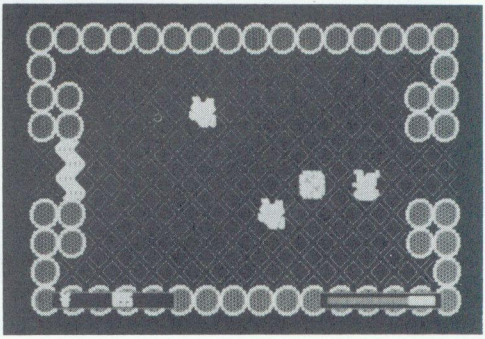
The graphics are quite good, and the game is a great deal of fun to play. One small quirk is that it is quite difficult to go up and down steps as you need to be in exactly the right position. Apart from that, I thoroughly enjoyed the challenge.



them with the block is better.

The packaging lures you to late night playing with the quest to reach the 'ultimate' seventh gem. However, there are yet more levels stretching upwards from there...

Quibbles like that aside, Labyrinth seems to have hit just the right balance between action, graphics, intrigue, and other memory gulping features to provide a very addictive game that will keep you up to the small hours with ease.



Letter Quality

Tim Powys-Lybbe puts the Ibico LTR-1 printer through its paces.

This is a remarkable little printer. It is not dot matrix, it is not daisy wheel, it is not golf ball. Instead it has five circles of 'rubber bands' with letters on them. These bands spin round at great speed, do a shuffle to find the right one and then a hammer presses the right band at the right point to print the right letter. There is no ribbon, just an ink roller which wets the bands as they roll over it.

This produces very acceptable results. A keen eye can just detect three very faint smudges from the bands on a full sheet of typing. However, uninformed you'd swear blind that the printout was from a daisy wheel. One could criticize the print for not being totally black, and the individual letters have a habit of falling slightly out of alignment, but it compares well with the output of far more expensive daisy wheel machines at my place of work.

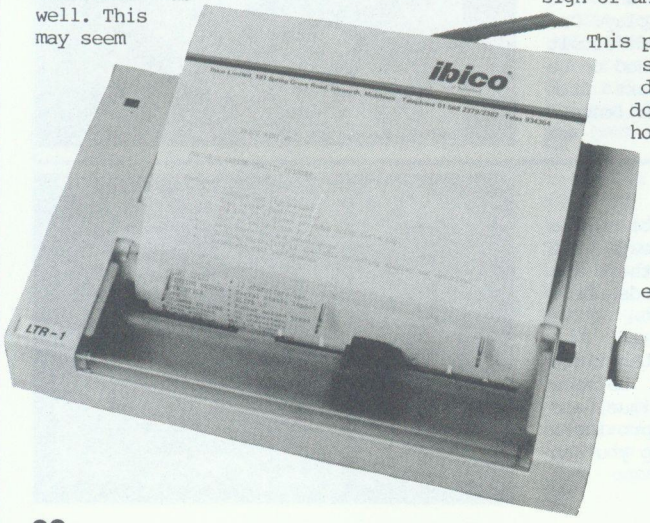
The LTR-1 is also cheap for that quality of result - it's available at about £180. It has a Centronics parallel interface and optionally there is a 1200 baud serial interface as well. This may seem too much to believe, but it's not. The only problem is that this is virtually all that the printer will do.

```
This  is a remarkable little
sy wheel, it is not golf ball.
ber  bands  with  letters on t
ed, do a shuffle to find the r
right band at the right point
```

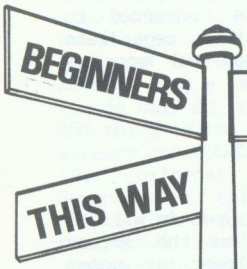
It will only accept A4 width sheets, not normal tractor-fed paper, though a printer roll could possibly be jury rigged at the back. It will not underline nor double strike. It will not form feed. It prints at a nominal 12 characters per second, perhaps a third of the speed of more expensive daisy wheels. There is only one character set and spacing of 12 characters per inch.

In use paper feeding is quite easy. This is just as well with its single sheet only capability. The ink roller is simple to change.

The obvious question is how long will it last? I did a soak test of printing off a 25 page document. I would like to have done 100 pages but at five minutes per page, plus the time lost when I didn't notice the bleep signal to change the page, this short test took nearly three hours. The results at the end were as good as those at the beginning. There was no sign of any degradation.



This printer is excellent for simple, short, typed (not dotted) documents. It would be ideal for domestic correspondence. If, however, you require fancy features such as underlining or bold print or you have lengthy documents to prepare, then this is not the printer for you. You would be better off with an (admittedly more expensive) daisy wheel printer. For its limited uses, though, there is no denying that the LTR-1 is excellent value for money and, on initial testing, it is most thoroughly recommended.



LOGIC

More about logic

Peter Lewis continues last month's theme by delving into some of the further intricacies of using logic in programs.

Last month I introduced some basic ideas about logic and dealt with some of the more unusual and perhaps unexpected uses of logic in Basic programs. We are now going to get a little more technical and detailed in pursuing some more useful and interesting applications.

First of all we need to learn some jargon. Keywords like AND, OR and NOT are called 'operators', and because of their nature, these are referred to as 'logical operators'. The complete set of logical operators in BBC Basic comprises AND, OR, EOR and NOT. NOT is an example of a 'unary' operator because it is always applied to a single value or variable, but all the others are examples of 'binary' operators because two values or operands are required.

BINARY

In order to really understand the use of logical operators you need to think in terms of binary numbers. This is not as frightening as it may sound.

Binary is a number system comprising only the digits 0 and 1. Computers use this system because they are built up from simple electronic components which themselves exist in one of two states, just like an on-off switch.

Most registers in the 6502 microprocessor and all memory locations comprise 8 bits (a bit is a binary digit) called collectively a byte. Converting between decimal and binary numbers is quite simple for small numbers, as shown in the table.

Decimal	Binary	Decimal	Binary
0	0	6	110
1	1	7	111
2	10	8	1000
3	11	9	1001
4	100	10	1010
5	101	11	1011

For other numbers, we need a rule to convert between decimal and binary. The basic principle is to divide a decimal number repeatedly by 2 recording the remainders to produce the binary equivalent. The converse of this, multiplying repeatedly by 2, is used to convert from binary to decimal.

If this sounds difficult, then why not use the computer to do the job for you. I have written two short functions to do the conversions. These could easily be incorporated into complete programs (this has been done for the magazine cassette/disc), or used in immediate mode. Since the main topic of this article is the use of logic, I will not dwell further on binary numbers and their conversion to or from decimal, but those who are interested

Decimal to Binary

```
1000 DEF FNbinary(decimal%)
1010 LOCAL bin$,dec%
1020 dec%=decimal%:bin$=""
1030 REPEAT
1040 bin$=CHR$(48+dec%MOD2)+bin$
1050 dec%=dec%DIV2
1060 UNTIL dec%=0
1070 =bin$
```

Binary to Decimal

```
1100 DEF FNdecimal(binary$)
1110 LOCAL dec%,i%
1120 dec%=0:i%=1
1130 REPEAT
1140 dec%=2*dec%+ASC(MID$(binary$,i%))-48
1150 i%=i%+1
1160 UNTIL i%>LEN(binary$)
1170 =dec%
```

should be able to follow the details from the two functions (I find it easier to treat the binary number as a string rather than as a pseudo-binary number since BBC Basic cannot handle binary numbers as such).

LOGICAL OPERATIONS

To understand how the logical operators work we need to construct a simple table. Because binary numbers consist exclusively of 1 and 0 the table also uses these two values. You can also think in terms of 1 representing TRUE and 0 representing FALSE. The table doesn't include NOT which has the simple effect of changing 1 to 0 or vice versa.

A	B	A AND B	A OR B	A EOR B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Let's see how we can relate this new information to our previous ideas of TRUE and FALSE. By typing in immediate mode we find, for example, that:

```
PRINT  3 OR 7   gives    7
PRINT 17 OR 5   gives   21
```

In order to make sense of this we must convert the numbers to binary and apply the results in the table for OR. We get:

```
3 is 00000011      17 is 00010001
7 is 00000111       5 is 00000101
     00000111 is 7       00010101 is 21
```

In each case the results are obtained by applying the results from the table to each pair of bits in turn to build up the answer.

USING LOGIC TO CHANGE CASE

Here's an example of a simple but very useful application of this kind of technique. If you've ever tried to write an input routine you will know the problems that can arise with upper and lower case input. Yet it is very easy to ensure that all input is converted to either one case or the other, regardless of the setting of the Caps-Lock key. Try the following:

```
100 REPEAT
110 PRINT CHR$(GET AND 223);
120 UNTIL FALSE
```

Any printable character typed in will be displayed in upper case on the screen irrespective of whether lower or upper case is entered. Try it and see. Alternatively, change line 110 to:

```
110 PRINT CHR$(GET OR 32)
```

You should now find that all input is echoed in lower case. You can understand why this works by looking at the ASCII codes (see User Guide page 486) for 'A' and 'a':

```
A is 65 (01000001 in binary)
a is 97 (01100001 in binary)
```

All ASCII characters are represented by a one byte code (8 bits in binary) and you can see that the two codes above differ only in the third bit from the left. This is not just true of the letter 'A' but of the whole alphabet. Forcing a 1 or a 0 in this position determines whether the character is lower case or upper case.

To convert to upper case we have to ensure that the third bit is 0. This is done in the program by ANDing the ASCII code of any character entered with the value 223 (11011111 in binary). Note the 0 in the third position. The value 223 used like this is called a 'mask'. Wherever the mask contains a 1, the resulting bit remains the same (as in the number being masked), and any 0 ensures that the bit in that position is set to zero.

To change any input to lower case, we must force a 1 in the third position, and this can be achieved in a similar way but using OR instead of AND. Masks are less common than some of the techniques described last month, but still very useful in the right place.

EXCLUSIVE OR

The exclusive OR operation, abbreviated to EOR, may seem a little unusual at first

sight, but has the most useful property that repeating an EOR operation a second time restores the original number. You can use this feature to code and decode messages by EORing each character in the message with a 'secret' code. The function FNcode will do this for any message. The odd thing is that if you then 'code' the already coded message the process restores the original text. The program demonstrates this using a code of your choice. Some codes you choose may produce unexpected results and you might like to work out why for yourself.

```
  10 REM Coding messages LOGIC23
 100 CLS
 110 PRINT'"CODING MESSAGES"
 120 INPUT'"Enter your coded message:"'
' message$
 130 INPUT'"Enter your secret code (1 t
o 255):" code%
 140 PRINT'"The coded message is:"
 150 A$=FNcode(message$,code%):PRINT'TA
B(10) A$
 160 PRINT'"and decoded again:"
 170 PRINT'TAB(10) FNcode(A$,code%)
 180 END
 190 :
1200 DEF FNcode(message$,code%)
1210 LOCAL i%,result$:result$=""
1220 FOR i%=1 TO LEN(message$)
1230 result$=result$+CHR$(ASC(MID$(mess
age$,i%)) EOR code%)
1240 NEXT i%
1250 =result$
```

The property of EOR restoring an original value is often used in graphics for animation. If you select Exclusive-OR plotting with the GCOL instruction then you can draw or plot a shape or character on the screen and yet, if you repeat the process, the object disappears leaving the background unaffected. The GCOL instruction allows you to AND, OR, or EOR the drawing colour with the background colour to produce a variety of different effects. In all cases, the logic table with this article will help you to work out the result of mixing foreground and background colours.

MISCELLANEOUS LOGIC

I want to conclude this discussion on logic with some more practical suggestions for using logic within your programs. If you look at many of the programs published in BEEBUG you will realise that we favour structured programs making frequent use of functions and procedures. A function always returns a value, a string or a number, but it could just as well be a logical value indicating the success or failure of the function. For example, if we define a function to search an array for a given value, we could write:

```
1000 DEF FNsearch(value)
1010 LOCAL found%:found%=FALSE
- - - - - -
1090 =found%
```

This assumes that the function sets found%=TRUE if the value being searched for is found. We could then write, in the main program:

```
100 IF FNsearch(248) THEN . . . .
```

There are many possibilities for this kind of use, and indeed the same idea can be applied to procedures, though you would need to use a 'global' variable as a procedure cannot return a value. A global variable is one that has the same meaning both within and outside a procedure.

Finally, a few words about the use of nested IF-THEN-ELSE statements. As you may know from reading the User Guide, BBC Basic has a quite sophisticated form of IF-THEN allowing several instructions, including further IF-THEN-ELSE statements to appear after the 'THEN' and the 'ELSE'. With even moderate nesting of IF-THEN-ELSE statements, confusion can easily arise because of the rather odd way in which BBC Basic handles this. Without going into a detailed analysis, there is a simple rule of thumb to keep you out of trouble:

```
'ELSE IF' is good practice
'THEN IF' is bad practice
```

So, if you structure a statement as:

```
    IF <condition> THEN <instruction>
ELSE IF <condition> THEN <instruction>
ELSE . . . . . .
```

you should not encounter any problems. If you start writing instructions like:

```
IF <condition> THEN
IF <condition> THEN <instruction>
               ELSE <instruction>
```

then don't be surprised if what you have written produces unexpected results.

GENERATING A PRINTER DRIVER FOR VIEW

This utility by Bernard Hill will enable all View wordprocessing enthusiasts to make the most of their printer's fancy features with this configurable printer driver.

The View word processing ROM is an excellent piece of software which is capable of many uses, but as the BBC micro is capable of driving many different types and styles of printer it needs a software 'patch' to drive the 'fancy' facilities of any given printer. This 'patch' is called a printer driver and consists of some machine code routines loaded between the addresses &400-&4FF. This program shows you how to create such a driver to your own specification and for your own printer. The driver may be stored using any filing system: cassette, disc, or even ROM. Once created and stored it is loaded by typing 'PRINTER <filename>' when in View command mode.

HOW TO USE A PRINTER DRIVER

A glance at the centre of the View function key strip reveals the existence of two HIGHLIGHT commands. The inclusion of the first of these characters in a document causes View to output an ASCII 128 to the printer driver when printing, and the second highlight generates an ASCII 129. If we can write a View driver which will interpret the 128 as "toggle underlining" (i.e. turn it on if it's off and off if it's already on) and ASCII 129 as "toggle bold text" then we can use these to generate simple underlining and bold headings in the printed document. Pressing the Highlight 1 key generates a character seen on the screen as underscore (_), but when printed with the correct driver the underlining mode is turned on until the next Highlight 1 character. A very similar thing happens with the Highlight 2 code, which appears as an asterisk (*). Thus what appears on the screen as:



This is a title

is printed as:

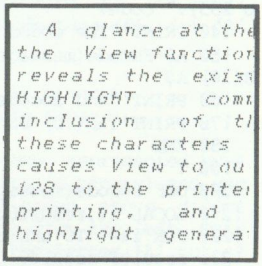
This is a title

Of course, the Highlight 2 key behaves similarly.

It might be supposed that we can only deliver two types of highlight to a document by this method but Acornsoft have thoughtfully allowed us to change the meaning of either or both of these highlight commands, even within the same document. Using the 'HT' edit command, Highlights 1 and 2 can be made to generate any ASCII character. For instance, the edit command:

HT 1 130

would mean that Highlight 1 now generates ASCII 130 (until switched to something else with another HT 1 command), and if our printer driver can be made to interpret ASCII 130 as an instruction to toggle Italic characters then we have a third possible highlight - always provided of course that your printer supports Italic characters!



Popular printers such as the Epson have a multitude of possible character formats such as Italic, condensed, bold, double-strike, enlarged, elite, superscript, superscript condensed etc, and it is a shame not to be able to use these from such a good word processing package as View.

This program amends this state of affairs and enables the user to choose his own highlight set - possibly even creating a whole family of drivers for specialised applications such as mathematical work, letterheadings, BEEBUG articles,...

For further details on the use of highlights and HT commands in View documents see the View Guide.

THE VIEW DRIVER GENERATOR - APPLICABILITY

If your printer's special functions are switched by the sending of non-printing ASCII characters (often using 27, the Escape character) and each change of function uses no more than 4 characters, then this program can be used, and up to 12 highlight styles can be embedded in one driver. The example used in the program listing is an example configuration for an Epson printer, where "Underline on" is generated by sending 27,45,1 and "Underline off" is 27,45,0 (corresponding in Wordwise to embedded command OC27,45,1, etc). The DATA statements at lines 1990-2000 reflect this fact. Note that the DATA statements must be padded out with zeros to make a total of 4 values. The strings heading these DATA lists ("Underline" and "Underline off") perform no useful function other than providing documentation of the driver for the user. The DATA statement at line 1980 is merely the total number of highlights used, in this case 11.

A search through your printer manual will reveal what character formats are available and what series of codes generate them. The DATA statements in lines 1990-2200 should be replaced by your own character formats and ASCII codes. Remember, though, that there must be exactly four per highlight (extend to four with non-functional zeros) and that the values in the DATA statements are in decimal (your manual may quote them in hex). The View documentation rather expects Highlight 1 to be underlining and Highlight 2 to be **bold** characters but it is entirely up to you.

```
129. If we can wi
driver which will
the 128 as
underlining" (i.e.
if it's off and
already on) and AS
"toggle bold text"
use these to gen
underlining and b
in the printed
Pressing the High
```

Remember also that when a change of character size (e.g. condensed) is envisaged then View's line formatting cannot be expected to work and a new ruler may need to be used for paragraphs in differing text sizes. Likewise, beware of centring double-size text without a new narrower ruler.

RUNNING THE PROGRAM

Those with BASIC I will need to read the section at the end first. As always with programs containing assembler sections, save the program before running it. The program is extensively commented (REM in Basic, "\" in assembler) but all these can be ignored when typing the program in from the keyboard.

When the program runs it asks for the name of the printer driver. This is the name of the file that it will be saved in. If an attached printer is enabled (with Ctrl-B) before the program is run, then a printout of the program documentation is automatically produced. The following documentation resulted from the program as given:

View Printer Driver : EPSON

Highlight Code	Function
128	Underline/Underline off
129	Bold/Bold off
130	Italic/Italic off
131	Double/Single Strike
132	Double/Single width
133	Condensed/Condensed off
134	UK/US chars
135	Superscript/Superscript off
136	Subscript/Subscript off
137	Elite/Pica
138	Proportional/Normal Spacing

The resulting driver has been tested with versions A1.4 and A2.1 of View.

PROGRAM WORKING

The documentation for version 1.4 of View indicates that the driver will be located at addresses &400-&4FF, but as this is required by the Basic assembler for workspace, it needs to be assembled elsewhere for running in page 4 - an ideal use for the new Basic II OPT instruction, which can take extended values of from 4 to 7, when code will be assembled at the address in O% for running at the address in P% (see lines 1050-1060). Another difference between a machine code program for a Basic application and this View application is

```
It might be supposed
n only deliver two t
ghlight to a docum
is method but Acorns
oughtfully allowed
ange the meaning of
both of these hi
mmands, even within t
cument. Using the 'H
mmand, Highlights 1
n be made to gener
```

that there is no known available location in zero page which can be safely used (Basic leaves &70-&8F for the user), and since one of the assembler instructions used needs zero page addressing, it is necessary to clear space by saving the contents on the stack and restoring later.

The View ROM expects the driver to start with a series of five JMP instructions at locations &400-&40E corresponding to the various functions the driver will have to perform. Only the first three are implemented in this driver, the last two being concerned with setting and reporting on micro-spacing. The first function (&400-&402 = JMP charout) is to output the character in the A register. If this character is less than ASCII 128 then it is simply sent to the screen (and hence to the printer if it has previously been enabled with the second function "printon" below). Should it be 128 or over it is a highlight code and a branch to a section of code (.highlite) is made. This code first checks on the status of that highlight - has it been enabled before? - if so, then send to the printer the 4 codes to turn it off, or if not, the 4 codes to turn it on. The relevant status byte is toggled at this stage.

Popular printers such as the Epson have a multitude of possible character formats such as Italic, condensed, bold, double-strike, enlarged, elite, super- script, superscript condensed etc, and it is a shame not to be able to use these from such a good word processing package as View.

The second JMP instruction (&403-405) must be to a section of code which initialises the printer. In this case it clears all the highlight status bytes and sends a VDU2 code to enable the printer.

state of affairs and enables the user to choose his own highlight set - possibly even creating a whole family of drivers for specialised applications such as mathematical work, letter-headings, BEEBUG articles, etc. For further details on the use of highlights and HT commands in View documents see the View Guide.

The third JMP instruction (&406-408) must be to disable the printer, a VDU3 is simply generated.

As they are not implemented, the other instructions must JMP to a sole RTS instruction.

DATA STORAGE

Immediately following the section of actual code are the data areas:

1. One byte temporary storage (reserved with the EQUB instruction).
2. Four bytes for each code to enable a highlight.
3. Four bytes for each code to disable a highlight.
4. One byte for each highlight status. 0=highlight off, 1=highlight on.

Areas 2 and 3 are poked in with the ? operator in lines 300 & 360, the status bytes requiring no prior setting but will appear at the end (line 1830).

PROGRAM DEVELOPMENT

The comments within the code should render it relatively easy to follow for anyone acquainted with machine code, and to modify further if required. If not, don't worry, the program works as it stands and its workings do not need to be understood.

One of the more obvious additions would be to allow the '£' sign on the Beeb's keyboard to generate a '£' sign on your printer. Similarly other characters could be treated as special cases by inserting additional code between lines 1210 and 1220 to trap the particular character and change it to the one required by the printer. To cope with the '£' sign on the Epson printer the following simple addition could be made to the printer driver program:

```
1212 CMP #96  \Beeb £
1214 BNE print\not a £
1216 LDA #35  \load Epson £
1218 .print   \and print next character
```

This basic idea can be extended or modified further to cope with other requirements.

BASIC I CONVERSION

The procedure OSCLI is not available in Basic I. Replace line 450 with:

```
 450 PROCoscli(instr$)
```

and add the following lines:

```
3000 DEFPROCoscli(x$)
3010 DIM x 80
3020 $x=x$
3030 X%=x MOD 256:Y%=x DIV 256
3040 CALL &FFF7
3050 ENDPROC
```

The pseudo-operator EQUB is not available in Basic I. Replace line 1790 with:

```
1790 .temp OPT FNequb(0)
```

and add:

```
3100 DEF FNequb(x):?P%=x:P%=P%+1:=opt
```

The higher values of OPT are not available. Alter lines 1050 and 1060 to:

```
1050 FOR opt=0 TO 2 STEP 2
1060 O%=codeaddress:P%=O%
```

Some relocation addresses will then need to be changed. Add the following lines:

```
1842 RESTORE 3200
1844 FOR n=1 TO 16
1846 READ x
1848 codeaddress?x=4
1849 NEXT n
3200 DATA 2,5,8,&B,&E,&11,&19,&34,&47
3210 DATA &4E,&55,&5B,&60,&65,&6B,&8A
```

Amend lines 300 and 360 to read:

```
300 ontable?(4*i+j)=x
360 offtable?(4*i+j)=x
```

```
10 REM Program ViewDriver
20 REM Version B1.1
30 REM Author Bernard Hill
40 REM BEEBUG July 1985
50 REM Program subject to copyright
60 :
100 MODE 7
110 codeaddress=&3400
120 REM code assembled here but will
130 REM be run at &400
140 max=12 : REM max no. of highlights
150 HIMEM=codeaddress
160 DIM onname$(max-1),offname$(max-1)
170 REM strings for documentation only
180 PROCassemble : REM assemble code
190 :
200 REM now site the data and
210 REM produce documentation
220 :
230 RESTORE 1980
240 READ n : REM no. of highlights
250 IF n>max THEN PRINT "Too many high
lights":END
260 FOR i=0 TO n-1
270 READ onname$(i)
280 FOR j=0 TO 3
290 READ x : REM the higlight ON codes
300 ontable?(4*i+j+codeaddress-&400)=x
310 REM put them in the table
320 NEXT j
330 READ offname$(i)
340 FOR j=0 TO 3
350 READ x : REM highlight OFF codes
360 offtable?(4*i+j+codeaddress-&400)=x
370 REM put in the table
380 NEXT j
390 NEXT i
400 INPUT "Give printer name : "P$
410 instr$="*SAVE "+P$+" "+STR$~codead
dress+" "+STR$~(codeaddress+255)+" 400 4
00"
420 REM SAVE assembled code + data
430 REM will have load address = &400
440 PRINTinstr$ : REM let's see what's
happening
450 OSCLI(instr$) : REM save code + da
ta on current filing system
460 PROCshowoptions : REM produce docu
mentation
470 END
480 :
1000 DEFPROCassemble
1010 table=&70
1020 REM a zero page location is needed
1030 oswrch=&FFEE
1040 osasci=&FFE3
1050 FOR opt=4 TO 6 STEP 2
1060 O%=codeaddress : P%=&400
1070 REM assembled at &3400 for later
1080 REM running at &400
1090 [ OPT opt
1100 JMP charout \print the char in (A)
1110 JMP printon \enable printer driver
1120 JMP printoff\disable
1130 JMP sethmi  \microspacing
                 - not implemented
1140 JMP retopt  \return options
                 - not implemented
1150 .charout    \write out (A)
1160 STA temp
1170 PHA:TXA:PHA:TYA:PHA
                 \save registers
1180 LDA temp    \and recover byte
                  to be printed
1190 CMP #128    \128 or over?
1200 BPL highlite\yes so its a
                  highlight code
1210 JSR osasci  \if not just print it
1220 .return
1230 PLA:TAY:PLA:TAX:PLA \restore regs
1240 RTS         \and go home
1250 .printon    \print initialisation
1260 LDA #2
1270 JSR oswrch  \send printer on char
1280 LDA #0      \and clear all high-
                 light status registers
1290 LDX #max-1  \with a simple
1300 LDA #0      \loop
```

```
1310 .loop
1320 STA highstats,X
1330 DEX
1340 BPL loop
1350 RTS
1360 .printoff    \terminate printing
1370 LDA #3       \send printer
1380 JSR oswrch   \off code
1390 .retopt      \not implemented
1400 .sethmi      \ditto
1410 RTS          \so just go back
1420 .highlite    \send highlight
1430 LDA table : PHA \save 2 zero page
1440 LDA table+1 : PHA \locations
                       on stack
1450 LDA temp     \load the char to be
                   printed
1460 SEC          \ready to subtract
1470 SBC #128     \to give offset into
                   highlights
1480 TAY          \use Y as offset
                   register
1490 LDA highstats,Y \is highlight set?
1500 BNE set      \yes ...
1510 LDA #1       \no,
1520 STA highstats,Y \so set it
1530 LDA #ontable MOD 256 \load address
                   of ON table
1540 STA table    \to the ZPG location
1550 LDA #ontable DIV 256
1560 STA table+1  \high byte too
1570 JMP output   \and send the 4 chars
1580 .set         \status was set
1590 LDA #0
1600 STA highstats,Y \so clear it
1610 LDA #offtable MOD 256\load address
                   of the OFF table
1620 STA table
1630 LDA #offtable DIV 256
1640 STA table+1
1650.output       \send the 4 chars
                   from the table
1660 TYA : ASL A : ASL A : TAY
                  \quaduple Y to get
                   the table offset
1670 LDX #4       \counter
1680.loop2
1690 LDA #1       \to printer only
1700 JSR oswrch
1710 LDA (table),Y \send 4 chars
1720 JSR oswrch
1730 INY
1740 DEX
1750 BNE loop2
1760 PLA : STA table+1
1770 PLA : STA table
                  \restore ZPG locations
1780 JMP return   \all done!
1790 .temp EQUB 0\odd storage location
1800 .ontable     \this is where the
                   highlight ON codes go
1810 ]
1820 offtable=ontable+4*max : REM and t
he highlight off codes are here
1830 highstats=offtable+4*max : REM fol
lowed by highlight status bits
1840 NEXT opt
1850 PRINT"Last address used ";~(highst
ats+max) : REM must not be over &4FF
1860 ENDPROC
1870 DEFPROCshowoptions
1880 CLS
1890 PRINT TAB(10);"View Printer Driver
: ";P$''
1900 PRINT"Highlight"'"Code","Function"'
1910 FOR i=0 TO n-1
1920 PRINT TAB(2);128+i;TAB(11);onname$
(i);"/"offname$(i)
1930 NEXT i
1940 ENDPROC
1950 :
1960  REM data section
1970 :
1980 DATA 11 : REM total no. of highlig
hts used
1990 DATA Underline,27,45,1,0
2000 DATA Underline off,27,45,0,0
2010 DATA Bold,27,69,0,0
2020 DATA Bold off,27,70,0,0
2030 DATA Italic,27,52,0,0
2040 DATA Italic off,27,53,0,0
2050 DATA Double,27,71,0,0
2060 DATA Single Strike,27,72,0,0
2070 DATA Double,27,87,1,0
2080 DATA Single width,27,87,0,0
2090 DATA Condensed,15,0,0,0
2100 DATA Condensed off,18,0,0,0
2110 DATA UK,27,82,3,0
2120 DATA US chars,27,82,0,0
2130 DATA Superscript,27,83,0,15
2140 DATA Superscript off,27,84,18,0
2150 DATA Subscript,27,83,1,15
2160 DATA Subscript off,27,84,18,0
2170 DATA Elite,27,77,0,0
2180 DATA Pica,27,80,0,0
2190 DATA Proportional,27,112,1,0
2200 DATA Normal Spacing,27,112,0,0
2210 DATA Half,27,65,6,0
2220 DATA Normal line spacing,27,50,0,0
```

Single Drive Disc Back-up

Alan Webster has produced another of his short disc utilities, a program to make life easier for all those users backing up their discs with but a single disc drive.

One of our members wrote to us wishing to know if there was an easy way to produce a back-up of side 0 of one disc on side 2 of a different disc using a single drive. This letter prompted me to write the following utility which will back-up any side of one disc to any side of another (single or dual discs).

When run, the program will first ask you the source drive number (a) and the destination drive number (b). It then asks whether a 40 or 80 track back-up is required. Once it has this information, it will proceed to back-up side a to side b. The program reads in ten tracks at a time from side a, then prompts the user to exchange discs, and then writes the ten tracks to side b of the second disc.

The back-up utility requires the area of memory from &1700 to &7100 to store the data from the disc, and therefore the utility needs to be run with PAGE set to &1200 first (PAGE=&1200 <return>).

PROGRAM NOTES

The main part of this program is the procedure PROCtracks(H). This sets up the OSWORD parameter block for disc control with A% set to &7F (see Advanced Disc User Guide page 209). The format of the block is as follows:

XY+0	Drive Number
XY+1-4	Memory Address of source or destination data
XY+5	Number of parameters
XY+6	Command Value
XY+7	Parameter 1
XY+8	Parameter 2
XY+9	Parameter 3

The command value is &4B to write a sector and &53 to read a sector. Parameter 1 is the track number, parameter 2 is the sector and parameter 3 is &20+number of sectors.

Please take extreme caution when using this OSWORD command if you modify this program at all, as it can wreck an entire disc full of data very easily.

```
  10 REM PROGRAM BACKUP
  20 REM VERSION B0.4
  30 REM AUTHOR  Alan Webster
  40 REM BEEBUG  July 1985
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 DIM BUF% 10:MODE7:@%=2
 110 CLS:PRINT'CHR$129"Program To Backu
p Side ";:D1=GET:VDUD1:PRINT" to Side ";
:D2=GET:VDUD2:PRINT'CHR$129"of another d
isc":D1=D1-48:D2=D2-48:IF D1<0 OR D1>3OR
 D2<0 OR D2>3 THEN VDU7:GOTO 110
 120 PRINT'CHR$134"40 or 80 track :*";
 130 REPEAT:VDU127:G=GET:VDUG
 140 UNTILG=52 OR G=56:VDU48
 150 G%=(G-48)*10-1:VDU28,0,24,39,10
 160 FORQ%=0 TO G% STEP10
 170 PROCdisc0:PROCtracks(0,D1)
 180 PROCdisc2:PROCtracks(2,D2)
 190 NEXT:PRINT''':END
 200 :
1000 DEFPROCdisc0:CLS
1010 PRINTCHR$130"Please insert Source
disc (reading side"CHR$130;D1;") and pre
ss any key.";:*FX21
1020 G=GET
1030 ENDPROC
1040 :
1050 DEFPROCdisc2:CLS
1060 PRINTCHR$131"Please insert Destina
tion disc (writing"CHR$131;"side ";D2;")
 and press any key.";:*FX21
1070 G=GET
1080 ENDPROC
1090 :
1100 DEFPROCtracks(H,D)
1110 X%=BUF% MOD 256:Y%=BUF% DIV 256
1120 ?BUF%=D:BUF%?5=3:BUF%?6=&53-(H*4)
1130 BUF%?9=&2A:A%=&7F:BUF%?8=0
1140 IFH=0PRINT''"Reading Track "; ELSE
PRINT'''"Writing Track ";
1150 FOR W%=Q% TO Q%+9
1160 PRINT W%;:VDU8,8
1170 BUF%!1=&1700+((W%-Q%)*2560)
1180 BUF%?7=W%:CALL &FFF1
1190 IF BUF%?10<>0 PROCerror
1200 NEXT
1210 ENDPROC
1220 :
1230 DEFPROCerror
1240 PRINTCHR$129"Disc Error ";~BUF%?10
1250 END
```

ACCELERATOR

Have Computer Concepts at last produced the Basic Compiler all Beeb users have been waiting for. Geoff Bains has taken a detailed look at this new product to see if the wait has been worthwhile.

Although BBC Basic is very fast, it's not fast enough for some people. The easiest way to speed up your programs, without having to learn a new language, is to turn the Basic program into machine code with a compiler. Whereas an interpreter converts each program statement into machine code as it is executed, a compiler will take the whole program and turn it into a complete machine code program, with all the inherent speed advantages. Computer Concepts Accelerator package is a compiler for the BBC micro for £64.40.

Accelerator comes on two ROMs, 16K and 8K, along with a manual, quick reference card, and a disc. Using Accelerator, Basic programs can be compiled either to an intermediate code called G-code or to standard 6502 machine code.

Compiling to G-code is the more efficient of the two, in both speed of compilation and length of the object code, but of course the resulting program requires a G-code interpreter to reside in the machine to run. Such a G-code interpreter is contained in the Accelerator ROMs but a G-code program is no use if you want to distribute your programs to friends.

Compiling to G-code has been made very simple. You simply type *ACCEL. and the compiler prompts you for the file name of the Basic program to compile. This can be already in the machine, or on disc. The object filename to store the compiled program under is then asked for (again the compiled program in G-code can be put straight into memory) and the program is compiled in three passes.

You can then re-run the G-code program again and again as desired. So typing *ACCEL. and pressing RETURN three times gives you a kind of supercharged RUN for your Basic program in memory. Alternatively G-code programs on disc can be run by typing *GRUN.

Any errors found during compilation will drop you back into Basic ready to edit your original program. You are quite likely to make 'errors' because of the quirks of G-code. Many Basic keywords are implemented strangely or not at all in the G-code compiler. The worst case is user functions. If a function is to return a string, then its name must have a $ on the end. That's understandable and not a worry. What is, however, is that the $ must only be placed on the end of the name when the function is called and not when the function is defined. In this way Accelerator requires you to write a program to be compiled in a way that prohibits it running in Basic - that's not a good idea.

G-CODE COMPILER. Basic keywords omitted or implemented differently.

CHAIN	FN	PAGE	TRACE
EVAL	LOMEM	RND	TOP

The 'Convert' program to produce stand-alone machine code programs, capable of being run on any BBC micro, is on the disc. A Basic program must first be compiled to G-code and stored on disc before being compiled to machine code. The machine code programs produced consist mostly of sets of subroutines copied directly from a library on the disc. There are three types of libraries available on the disc offering greater or lesser subsets of Basic with the corresponding greater or lesser length of the resulting machine code program. The machine code program is liable to be around three times the length of the original Basic one.

None of the libraries offer the full set of Basic keywords. The worst restriction is that the converter cannot handle real numbers. Only integers are allowed and these are stored to 16 bit accuracy only (+ or - 32767) instead of the 32 bits (+ or - 2147483647) used by Basic. This causes considerable

MACHINE CODE CONVERTER. Basic keywords omitted or implemented differently.

ACS	EVAL	ON GOTO	SIN
ASN	FN	ON GOSUB	SQR
ATN	GOTO	PAGE	TAN
COS	INT	PI	TOP
CHAIN	LN	RAD	TRACE
DEG	LOG	RESTORE	/
EXP	LOMEM	RND	

difficulties. For example a Basic statement such as:

```
IF (A%+B%) > 40000 THEN PRINT "big" ELSE PRINT "small"
```

when compiled, with values of 30000 and 15000 for A% and B%, would print "small" because 45000 is taken as a negative number (-10535). The same problem plagues ADVAL values, !, memory addresses, in fact any number greater than 32767, although any number specified in the program as a hex number (with a &) is converted properly. All this makes writing a Basic program suitable for compiling to machine code not far off learning a new language.

Escape is also disabled in the compiled program and any need for this has to be catered for specially by testing for that key by reading the Escape key flag at &FF. Any errors in the original program are not fully reported so it is essential that it is fully debugged before compilation.

An excellent bonus for the Converter, however, is that it can compile your program into machine code in a sideways ROM format.

Given the restrictions on the source code of both compilers, the resulting programs do show a worthwhile increase in speed. The gain in speed will depend on the type of program. The most dramatic increases come from the storage of variables. In the compiled program (machine code or G-code) variables are referred to by address, not by name as they are in Basic, and so they are accessed much faster. A simple addition of two variables requires only the addition and none of the rigmarole of fetching the values of the two variables. You won't find graphics commands going much faster, when compiled, as the Basic graphics commands just access machine code routines in the OS anyway. So a statement such as

```
GCOL0,3:MOVE 100,100:DRAW 1000,1000
```

will take almost exactly the same time whether compiled or not. However if a large number of variables are involved:

```
FOR X%=0 TO 100
Y1%=X%*5
Y2%=X%*10
MOVE X%,Y%:DRAW X%,Y%
NEXT
```

then the compiled version will run faster.

The following program to draw circles takes 10.23 seconds run in Basic, 6.63 in its G-code version and only 4.63 seconds when compiled all the way to machine code.

```
 10 MODE 4
 20 VDU29,640;512;
 30 TIME=0
 40 FOR R%=50 TO 500 STEP 50
 50   X%=0:Y%=R%
 60   D%=3-2*R%
 70   REPEAT
 80     PLOT 69,-X%,-Y%:PLOT 69,-Y%,-
X%
 90     PLOT 69,X%,-Y%:PLOT 69,-Y%,X%
100     PLOT 69,X%,Y%:PLOT 69,Y%,X%
110     PLOT 69,-X%,Y%:PLOT 69,Y%,-X%
120     IF D%<0 THEN D%=D%+4*X%+6 ELS
E D%=D%+4*(X%-Y%)+10:Y%=Y%-4
130     X%=X%+4
140     UNTIL X%>Y%
150   NEXT R%
160 PRINT TIME
170 END
```

The Accelerator manual explains all the restrictions it suffers from and the underlying principles for the most part clearly. However, these restrictions do severely limit its use. It is true that the G-code compiler produces efficient, fast programs but this is largely irrelevant if it is machine code that we really want. Without a larger subset of Basic available for the machine code converter, and without real number, and full integer arithmetic the machine code compiler has clear limitations. As it stands, Accelerator is the best compiler around for the BBC micro at the moment. If, however, you are hoping that it will solve all your problems in a flash, you are going to be disappointed.



ADVENTURE GAMES ADVENTURE GAMES AD

Two packages materialized inside my electronic crystal ball this month - a summons from the palace to deal with an evil dwarf, plus an adventure writing package for schools.

Title	: The Greedy Dwarf
Supplier	: Goldstar 1-2 Henrietta St., London, WC2E 8PS.
Price	: £9.95 cassette

Arfa, the greedy dwarf, has stolen the King's jewels and hidden them around the castle. This game's plot and location is set in a standard 'Castle and Dragon' land, with our old friend the torch to light the way; but it's still bereft of a long-lasting, copper-coloured filling!

A very nice feature of this game is that I managed to beat it! Special features involved include the use of function keys programmed to enable single key strokes for the most commonly used commands, and cursor keys which have been redefined to be used as movement controls. These soon proved to be a novel and ingenious method for tramping the dungeons. The game is text only but is none the worse for that. I found the traps and puzzles extremely satisfying as they are quite logical and I was able to repeatedly congratulate myself for being brilliant! A peek inside the command analyser revealed an extensive vocabulary, including a few words that I didn't think nice adventurers knew!

Some of the commands must be carried out 'QUICKLY' or 'QUIETLY' to escape the wrath of the guardian monsters, but you will soon learn to move quickly when a Rat is gnawing your finger. I found a minor bug with the EXAMINE command which cheered me up - I do love finding other author's mistakes! Should you forget to specify what object you wish to examine, the resulting message will reveal a small

```
You're at the foot of the staircase.
Passages lead north, east and west.

: N

You're in a long bending passage which
runs south and north-west.  The floor
is covered with dust and rubble.

: NW

You're in a large natural cavern where
every sound you make seems to echo for
ever.  Huge stalactites hang above you
and passages lead off all round.

: N

The noise of your footsteps makes a
stalactite break off and strike you
dead.

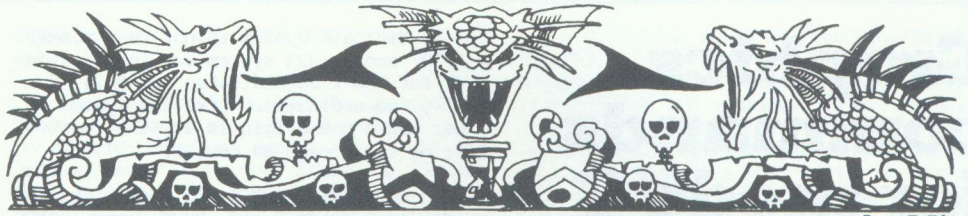
Would you like to start again? (Y/N)
```

secret. This is a very interesting game, none too hard, which deserves more recognition.

Title	: The Last Adventure
Supplier	: Learning and Training Systems Ltd. Alcester Rd., Studley, Warwickshire.
Price	: £17.95 disc (£14.95 schools)

An adventure generator is a program which is designed to take the blood, sweat and tears out of programming your own adventures. Writing the storyline of an adventure and designing the traps and problems is one aspect which most mortals can cope with, but actually programming the beast is a very different can of dragons.

This package is aimed mainly at educationalists who wish to quickly and simply create adventures for use by school children. For this reason the generator has been made easy to use but as a consequence, the results are less than riveting. The game produced may be played with either a Concept keyboard or the



by Mitch

ENTURE GAMES ADVENTURE GAMES ADVEN

normal function keys, which makes it attractive to the very young or the physically handicapped. Normal movement commands of NORTH, SOUTH, etc., are replaced by simply pressing the key which corresponds to the location the player wishes to move to. The generator permits the writer to specify problems and solutions for each location, and where required, entry conditions which must be satisfied before the player can move to that area.

In addition to copious notes and suggestions, a sample game is supplied to demonstrate the use of the package. The game tends to give multiple choice questions, requiring the answer a, b or c. To test this game, and hence the generator, I replied "ELEPHANT" to see if the game trapped this as an illegal answer - it didn't! Throughout, the game tends to assume that if you do not reply YES you must have said NO. When dealing with children, and other evil dwarfs, sloppy logic is not sufficient. A child could type rubbish indefinitely to this game, and both game and player would continue happily for a very long time. It should also be pointed out that the generator does not produce 'stand-alone' games, but a data file which must be run in conjunction with the generator package. This means that you cannot pass a copy of your masterpiece to your friends or colleagues unless they also possess a copy of the generator.

As an alternative to adventures the package can be used to recreate simulations of particular environments - e.g. visits to nature reserves.

For those wizards looking for a more commercial generator, I note that a BBC version of 'The Quill' is shortly to be released. This game writer which has been available for a year on the Spectrum (pardon me while I spit!), does produce stand-alone, machine code games, which are already being marketed.

For the Spell Book this month a Kentish wizard has sent me the following trick to confuse those trolls who would attempt to copy your game files. This involves saving and loading using filenames made up of the alternative mode 7 character set.

As can be seen from pages 488 and 489 of the User Guide, the letters A - Z appear twice in this character set. This means that either PRINT CHR$(65) or PRINT CHR$(193) will print the character 'A'. Pressing the standard keyboard 'A' character corresponds to CHR$(65) and it is this set which the filing system expects. To save a file called ABC using the alternative letters, use the following:

```
NAME$ = CHR$(193)+CHR$(194)+CHR$(195)
SAVE NAME$
```

The file cannot now be reloaded by simply typing LOAD "ABC".

To confuse the trolls, you should ensure that your main adventure program is saved with an alternative filename in this way. The header program should contain a CHAIN command which will appear, when listed, as CHAIN "ABC", or whatever. However, while the characters 'ABC' will be seen as the filename, you should ensure that it is the values of &C1, &C2 and &C3 (decimal 193, 194 and 195), which have been poked into the correct position in the file for 'ABC'.

To achieve this, insert the command CHAIN "ABC" into the header program and then examine the memory using a monitor utility such as EXMON to find the position of the letters (codes &41, &42 and &43 in this case). Using these addresses, change the contents using the ? operator (e.g. ?&ØEFØ=&C1).

Graphics commands for mode 7

If you thought that plotting graphics was limited to the so called graphics modes then think again. G. Middleton describes how to use MOVE and DRAW to plot graphics in mode 7 as well, and so much easier too.

Many Beeb users do not make the most of mode 7. Although it takes up the least memory of all of the eight display modes on the BBC, the very limited graphics control available in this mode puts people off. Mode 7 in fact has a graphics resolution of 78 by 75 covering most of the screen (with a single column of control characters taking up the left hand side). However, this is normally very hard to use because you cannot simply plot a dot using the PLOT command or draw a line using the DRAW command but must print whole graphics characters. This program enables you to use the PLOT, DRAW and MOVE commands in mode 7 almost exactly as you can in the other graphics modes.

The program adds these commands to those normally available in mode 7. Once the program has been run, your own Basic program can include PLOT, DRAW and MOVE in the normal way but in mode 7. The following commands are added to the mode 7 repertoire.

PLOT 1,x,y - plots a single point at co-ordinates x,y.

PLOT 0,x,y - erases a point at co-ordinates x,y.

MOVE x,y - moves the graphics cursor to x,y.

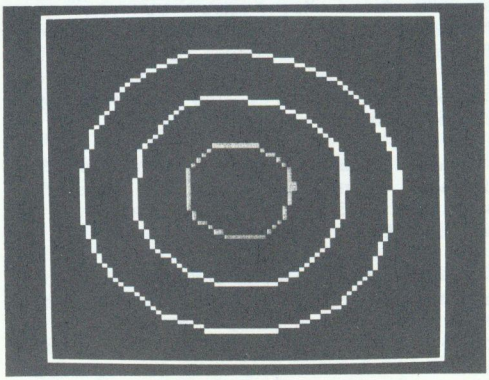
DRAW x,y - draws a line from the last position of the graphics cursor to the point x,y.

The co-ordinate system is not the normal 0-1279 and 0-1023 system but is based on 0-77 for the x co-ordinates and 0-74 for the y co-ordinates. Like the system in other modes the origin is at the bottom left of the screen as normal.

In addition, PLOT 4 and PLOT 5 can be used in place of MOVE and DRAW respectively, as normal. When you are drawing or plotting you will find that if you try to draw or plot anything off the screen then the routine will simply ignore the co-ordinates. Nor will the routine allow you to plot over the coloumn of graphic control codes at the left of the screen.

This routine can only directly produce graphics in one colour, because of the nature of the mode 7 display. The graphics created by this routine are made up of teletext graphic characters, selected to light the appropriate pixel at the points required.

To display the teletext graphics characters the screen must also have graphics colour codes to the left of the characters. Suitable codes are placed all the way down the left hand side of the screen by PROCscreen. This procedure must be called (with the required colour number for the graphics as its parameter) before any graphics are plotted on the screen. In this way only one colour graphics can be displayed on the screen at a time. However, by surreptitious placing of other graphics colour codes around the screen, different parts of your picture can be made to appear in different colours. The demo procedure

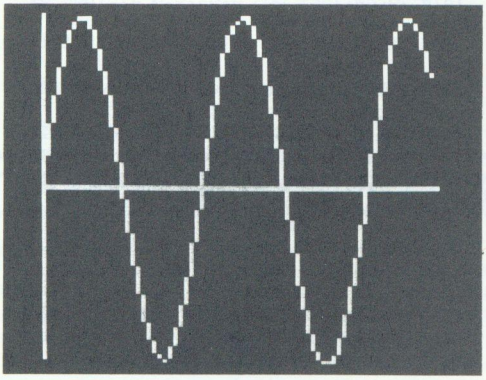


shows how this can be done (see line 2620).

The short demonstration procedure draws a frame around the screen and concentric circles inside this all in yellow. Note that the frame is drawn without using the bottom right character position. This is to stop the display scrolling and removing the control codes on the left hand side - with disastrous effects in the graphics. This could, however, be prevented by poking zero to location &ØD. The innermost circle is drawn in cyan by printing cyan graphics codes to the left of it and yellow codes to the right to restore the other circles to yellow.

When you have typed in the program, save it before you run it. The program places the machine code in memory at &7800, just below the mode 7 screen. If you want to change this position (for example if your program uses other modes as well as mode 7 that will overwrite this memory area) then alter line 1010. The code takes up &304 locations.

The program listed here gives you the option of saving the machine code alone which is desirable for use by your own programs. When prompted, type 'Y' and enter a filename for the code. In future when you want to use the plot routine simply load in the machine code by typing *RUN <filename>, either in immediate mode or as a command from within your own program. The machine code routine is protected against any form of Break so you only ever have to load it in once.



The routine works by intercepting the VDU extension vector at &226 (see the Advanced User Guide page 261). Unrecognized VDU commands (i.e. graphic commands in mode 7) are passed via this vector to the routine which checks to see if the command is a PLOT Ø, 1, 4, or 5 and pokes the required characters onto the screen.

The assembler section of the program is liberally sprinkled with comments to help you to understand what is going on. However, there is no need to know anything about the inner workings of the program if you don't want to. You can just use it.

```
  10 REM Program Mode 7 graphics
  20 REM Author  G.Middleton
  30 REM BEEBUG  July 1985
  40 REM Program subject to copyright
  50 :
  60 ON ERROR GOTO 2650
  70 :
 100 MODE7
 110 PROCassemble
 120 PROCscreen(3)
 130 PROCdemo
 140 dummy=GET
 150 END
 160 :
1000 DEF PROCassemble
1010 code%=&7800
1020 IF ?&8015=50 THEN ifpa=&A2BE:apack
=&A38D:aunp=&A3B5:fpil=&A3E4:adiv=&A6AD:
amult=&A656
1030 IF ?&8015=49 THEN ifpa=&A2AF:apack
=&A37E:aunp=&A3A6:fpil=&A3F2:adiv=&A6B8:
amult=&A661
1040 IF ?&8015<>49 AND ?&8015<>50 THEN
PRINT"Unrecognizable Basic":END
1050 plotx=&76:ploty=&77:presentx=&311:
presenty=&313
1060 command=&78:x=&7B:y=&7F:e=&7C
1070 osbyte=&FFF4:oswrch=&FFEE
1080 oldx=&310:oldy=&312
1090 *FX 247,0
1100 *FX 248,0
1110 *FX 249,0
1120 lookup%=code%
1130 !lookup%=&08040201:lookup%!4=&4010
1140 code%=code%+10
1150 FOR pass=0 TO 3 STEP 3
1160 P%=code%
1170 [OPT pass
1180 .start
1190 STA command
```

```
1200 CMP#0:BEQ plot
1210 CMP#1:BEQ plot
1220 CMP#4:BEQ plot
1230 CMP#5:BEQ plot
1240 RTS
1250 :
1260 \ If it is neither PLOT 0,X,Y nor
PLOT 1,X,Y then return to BASIC
1270 :
1280 .plot
1290 LDA &320:STA plotx:STA presentx:LD
A &322:STA ploty:STA presenty
1300 LDA &318:STA &79:LDA &319:STA &7A
1310 LDA &321:BEQ ok:RTS
1320 .ok LDA &323:BEQ ok1:RTS
1330 .ok1 LDA plotx:CMP#77:BEQ ok2:BCC
ok2:RTS
1340 .ok2 LDA ploty:CMP#74:BEQ ok3:BCC
ok3:RTS
1350 :
1360 \ If the coordinates are out of ra
nge then return to BASIC
1370 :
1380 .ok3 LDA command:CMP#5:BNE notdraw
:JMP draw
1390 .notdraw CMP#4:BNE startplot:JMP m
ove
1400 .startplot LDX presentx:STX oldx:L
DY presenty:STY oldy
1410 LDX plotx:LDY ploty
1420 STX &75
1430 LDA#&FF:STA &74
1440 LDA#31:JSR oswrch
1450 STX &70:LDA#2:STA &71
1460 JSR divide:INC &70
1470 LDA &70:JSR oswrch
1480 STY &70:LDA#3:STA &71
1490 JSR divide
1500 SEC:LDA#24:SBC &70:JSR oswrch
1510 LDA &75:AND #1:STA &73
1520 SEC:LDA#2:SBC &72:ASL A
1530 CLC:ADC &73:TAX:LDA lookup%,X:STA
&73
1540 :
1550 \ Work out the the value to be pri
nted on the screen
1560 :
1570 LDA command:CMP#0:BNE pplot:SEC:LD
A#&FF:SBC&73:STA &74
1580 :
1590 \ If it was PLOT 0,X,Y then delete
 the point otherwise plot the point
1600 :
1610 .pplot LDA #135:JSR &FFF4:TXA
1620 ORA &73:ORA #128:AND &74:JSR oswrc
h
1630 LDA#31:JSRoswrch:LDA&79:JSRoswrch:
LDA&7A:JSRoswrch:RTS
1640 :
1650 \ End of plotting routine
1660 :
1670 .draw SEC:LDA plotx:SBC oldx:CMP#0
:BNE notline:JMP vline
1680 .notline SEC:LDA ploty:SBC oldy:CM
P#0:BNE notline2:JMP hline
1690 .notline2
1700 SEC:LDA ploty:SBC oldy:STA &2A:JSR
 fillint
1710 JSR ifpa:LDA#o MOD 256:STA &4B:LDA
#o DIV 256:STA &4C:JSR apack
1720 SEC:LDA plotx:SBC oldx:STA &2A:JSR
 fillint
1730 JSR ifpa:LDA#a MOD 256:STA &4B:LDA
#a DIV 256:STA &4C:JSR apack
1740 LDA#o MOD 256:STA &4B:LDA#o DIV 25
6:STA &4C:JSR adiv
1750 LDA#g MOD 256:STA &4B:LDA#g DIV 25
6:STA &4C:JSR apack
1760 JSR fpi1:LDA &2A:SEC:SBC#1:BEQ oth
ergrad:BPL othergrad:LDA &2A:CLC:ADC#1:B
MI othergrad:BEQ othergrad
1770 LDA plotx:CMP oldx:BPL bigger3:LDA
 plotx:STA x:LDA oldx:STA e:INC e:JMP dr
awline3
1780 .bigger3 LDA oldx:STA x:LDA plotx:
STA e:INC e
1790 .drawline3 LDA oldx:STA &7D:LDA ol
dy:STA &7E:LDA#1:STA command:.repeat3
1800 SEC:LDA x:SBC &7D:STA &2A:JSR fill
int
1810 JSR ifpa:LDA#g MOD 256:STA &4B:LDA
#g DIV 256:STA &4C:JSR amult
1820 JSR fpi1:LDA &2A:CLC:ADC &7E:STA p
loty:LDA x:STA plotx:JSR startplot:INC x
:LDA x:CMP e:BNE repeat3
1830 RTS
1840 :
1850 .othergrad LDA ploty:CMP oldy:BPL
bigger4:LDA ploty:STA y:LDA oldy:STA e:I
NC e:JMP drawline4
1860 .bigger4 LDA oldy:STA y:LDA ploty:
STA e:INC e
1870 .drawline4 LDA oldx:STA &7D:LDA ol
dy:STA &7E:LDA#1:STA command:.repeat4
1880 SEC:LDA y:SBC &7E:STA &2A:JSR fill
int
1890 JSR ifpa:LDA#spare MOD 256:STA &4B
:LDA#spare DIV 256:STA &4C:JSR apack
1900 LDA#g MOD 256:STA &4B:LDA#g DIV 25
6:STA &4C:JSR aunp:LDA#spare MOD 256:STA
 &4B:LDA#spare DIV 256:STA &4C:JSR adiv
1910 JSR fpi1:LDA &2A:CLC:ADC &7D:STA p
lotx:LDA y:STA ploty:JSR startplot:INC y
:LDA y:CMP e:BNE repeat4
1920 RTS
1930 :
1940 .vline LDA ploty:CMP oldy:BPL bigg
er:LDA ploty:STA x:LDA oldy:STA e:INC e:
JMP drawline
```

```
 1950 .bigger LDA oldy:STA x:LDA ploty:S
TA e:INC e
 1960 .drawline LDA#1:STA command:.repea
t LDA x:STA ploty:JSR startplot:INC x:LD
A x:CMP e:BNE repeat:RTS
 1970 .hline LDA plotx:CMP oldx:BPL bigg
er2:LDA plotx:STA x:LDA oldx:STA e:INC e
:JMP drawline2
 1980 .bigger2 LDA oldx:STA x:LDA plotx:
STA e:INC e
 1990 .drawline2 LDA#1:STA command:.repe
at2 LDA x:STA plotx:JSR startplot:INC x:
LDA x:CMP e:BNE repeat2:RTS
 2000 :
 2010 \ End drawing routine
 2020 :
 2030 .move LDA presentx:STA oldx:LDA pr
esenty:STA oldy:RTS
 2040 :
 2050 \ End of move routine
 2060 :
 2070 .fillint AND #128:BEQ fillzero:LDA
#&FF:JMP fill:.fillzero:LDA#0:.fill STA
&2B:STA &2C:STA &2D:RTS
 2080 :
 2090 .divide LDA #0:LDX #8
 2100 .loop ASL &70:ROL A:CMP &71:BCC le
ss:SBC &71:INC &70
 2110 .less DEX:BNE loop:STA &72:RTS
 2120 :
 2130 \ End of divide routine
 2140 :
 2150 .setup
 2160 LDA#start MOD 256:STA &226:LDA#sta
rt DIV 256:STA &227
 2170 RTS
 2180 .set
 2190 LDA#247:LDX#&4C:LDY#0:JSR osbyte:L
DA#248:LDX#setup MOD 256:JSR osbyte:LDA#
249:LDX#setup DIV 256:JSR osbyte:JMP set
up
 2200 :
 2210 \ End of set up routine
 2220 :
 2230 .a OPT FNclear
 2240 .o OPT FNclear
 2250 .g OPT FNclear
 2260 .spare OPT FNclear
 2270 :
 2280 ]NEXT pass
 2290 CALL set
 2300 PRINT'CHR$130;"Do you want to save
 the machine code?";CHR$131;
 2310 REPEAT awn$=GET$:UNTIL awn$="Y" OR
 awn$="N"
 2320 IF awn$="Y" OR awn$="y" THEN PROCs
ave
 2330 ENDPROC
 2340 :
 2350 DEF FNclear
 2360 !P%=0
 2370 ?(P%+4)=0
 2380 P%=P%+5
 2390 =pass
 2400 DEF PROCsave
 2410 DIM os% 30
 2420 PRINT'CHR$130;"Filename ";CHR$131;
:INPUT name$
 2430 $os%="SAVE "+name$+" "+STR$~(looku
p%)+" "+STR$~(P%)+" "+STR$~(set)
 2440 PRINTCHR$130;$os%
 2450 X%=os%:Y%=os% DIV 256:CALL &FFF7
 2460 ENDPROC
 2470 :
 2480 DEF PROCscreen(colour)
 2490 VDU12,&90+colour:FOR I=1 TO 24:VDU
 10,13,&90+colour:NEXT
 2500 VDU30
 2510 ENDPROC
 2520 :
 2530 DEF PROCdemo
 2540 MOVE 0,0:DRAW 75,0
 2550 DRAW 75,74:DRAW 0,74:DRAW 0,0
 2560 FOR R%=10 TO 30 STEP 10
 2570 MOVE 37+R%,37
 2580 FOR A=0 TO 7 STEP 0.3
 2590 DRAW 37+(R%*COS(A)),37+(R%*SIN(A))
 2600 NEXT A
 2610 NEXT R%
 2620 FOR Y%=9 TO 15:PRINT TAB(12,Y%) CH
R$150;:PRINT TAB(27,Y%) CHR$147;:NEXT Y%
 2630 ENDPROC
 2640 :
 2650 ON ERROR OFF
 2660 MODE 7
 2670 IF ERR<>17 REPORT:PRINT " at line
";ERL
 2680 END
```

HINTS HINTS HINTS HINTS HINTS HINTS

PROBLEMS WITH SIDEWISE - P.J. Elleray

Users with a 'Sidewise' expansion ROM board fitted may find problems with the keyboard. This only happens with a few Beebs and is due to contact between the OS ROM in the Sidewise and one of the keyboard board chips above it. Heavy typing can produce an intermittent contact here. A small piece of insulating material stuck to the bottom of the keyboard at this point should cure it.

BEEBUG Workshop

This month, Surac looks at a variety of useful ideas on drawing fast circles, essential in many graphics applications.

The Beeb's graphics are better than those on most micros and you probably use them at least some of the time. When you do, you quickly come up against the fact that, regrettably, there isn't a command to draw circles automatically. The obvious answer is to write a procedure to do the job and, this month, I'll show you an effective way of doing this.

Like lots of things in life, the "obvious" way of drawing a circle is not necessarily the best. I have listed here a very simple circle-drawing procedure.

It draws a circle with centre at (x%,y%) and radius r% graphics units. It works perfectly well, but it is terribly s-l-o-w. The main reason is that it has to work out no fewer than 72 sine and cosine functions in drawing the circle. Although BBC Basic is generally very fast, its trig calculations are not its best feature.

So, any routines which use a lot of trigonometry can be irritatingly slow. There are ways around that and here's one approach to drawing circles without calculating a lot of sines and cosines. It relies on two equations you may remember from school:

```
Sin(A+B)=Sin(A)*Cos(B)+
         Cos(A)*Sin(B)
Cos(A+B)=Sin(A)*Sin(B)-
         Cos(A)*Cos(B)
```

SIMPLE CIRCLE

```
10000 DEF PROCcircle(x%,y%,r%)
10010 LOCAL ang,steps%,xtemp%,ytemp%
10020 steps%=36
10030 MOVE x%,y%+r%
10040 FOR ang=2*PI/steps% TO 2.01*PI
      STEP 2*PI/steps%
10050   xtemp%=x%+r%*SIN(ang)
10060   ytemp%=y%+r%*COS(ang)
10070   DRAW xtemp%,ytemp%
10080   NEXT
10090 ENDPROC
```

Given the values of sine and cosine for a starting angle and for the fixed angle we want to step in, these equations make it easy to calculate a whole series of trig functions. The procedure above uses steps of 10 degrees and calculates sines and cosines for 10, 20, 30 degrees, etc. But:

```
Sin(10)=Sin(0+10)
Sin(20)=Sin(10+10)
Sin(30)=Sin(20+10)
Sin(40)=Sin(30+10)
etc.
```

Look at the equations above. If we know the sine and cosine of 0 and 10 degrees, we can calculate the values for 20 degrees. Having got there, 30 degrees follows, and then 40 degrees. So it goes - calculate the starting point and the increment (10 degrees in this case) and, from then, we need use only simple arithmetic to find the sine and cosine for every step up to 360 degrees, or as far as we want. It's not a very good way of finding, say, just Sin(257), but it's perfect for a regular series of calculations, and much faster than using the built-in trig functions.

If you're still with me, here's how to use this trick to draw a circle. It does the same job as PROCcircle, plus a bit more. If "fill%" is TRUE, then it will draw a filled circle (i.e. a solid disc), rather than a line. It also speeds things up by only calculating a quarter of the circle and drawing the 4 quadrants simultaneously.

Lines 10020-10040 choose the number of steps in drawing the circle - big ones

need more accuracy than little ones. Lines 10050 and 10060 calculate the fixed trig increments to use in the equations and line 10080 sets the initial values of Sin(0) and Cos(0). The "xo1%", "yo2%", etc., are used to record the progress of the 4 arcs.

```
                FAST CIRCLE
10000 DEF PROCfastcirc(x%,y%,r%,fill%)
10010 LOCAL a%,cinc,cos,plotno%,sin,
      sinc,stemp,steps%,xo1%,xo2%,
      xtemp%,yo1%,yo2%,ytemp%
10020 steps%=32
10030 IF r%>300 THEN steps%=40
10040 IF r%<50 THEN steps%=20
10050 sinc=SIN(2*PI/steps%)
10060 cinc=COS(2*PI/steps%)
10070 IF fill% THEN plotno%=85 ELSE
      plotno%=5
10080 sin=0:cos=1
10090 xo1%=x%:yo1%=y%+r%
10100 xo2%=x%:yo2%=y%-r%
10110 xo3%=x%:yo3%=y%+r%
10120 xo4%=x%:yo4%=y%-r%
10130 FOR a%=1 TO steps% DIV 4
10140   stemp=sin
10150   sin=sin*cinc+cos*sinc
10160   cos=cos*cinc-stemp*sinc
10170   xtemp%=r%*sin:ytemp%=r%*cos
10180   MOVE xo1%,yo1%
10190   IF fill% THEN MOVE x%,y%
10200   xo1%=x%+xtemp%:yo1%=y%+ytemp%
10210   PLOT plotno%,xo1%,yo1%
10220   MOVE xo2%,yo2%
10230   IF fill% THEN MOVE x%,y%
10240   xo2%=x%-xtemp%:yo2%=y%-ytemp%
10250   PLOT plotno%,xo2%,yo2%
10260   MOVE xo3%,yo3%
10270   IF fill% THEN MOVE x%,y%
10280   xo3%=x%-xtemp%:yo3%=y%+ytemp%
10290   PLOT plotno%,xo3%,yo3%
10300   MOVE xo4%,yo4%
10310   IF fill% THEN MOVE x%,y%
10320   xo4%=x%+xtemp%:yo4%=y%-ytemp%
10330   PLOT plotno%,xo4%,yo4%
10340   NEXT
10350 ENDPROC
```

Lines 10140-10160 are the important ones, actually calculating the sines and cosines by the equations above. The rest of the FOR loop is concerned with drawing the 4 arcs plus, if necessary, filling in the segments to the circle's centre to make a solid disc.

To give you an idea of the routine's speed, it draws circles about 3.5 times faster than PROCcircle. Because PLOT85 is inherently slow, it is not quite so good at discs - its speed drops to around 1.8 times. Overall, though, circles are drawn in fractions of a second - try it.

The best way to find out what the procedure can and can't do is to play with it, and to incorporate it in your own programs. However, here is a demo program to give you an idea of the routine's speed:

```
100 MODE 2
110 REPEAT
120   *FX 15,0
130   X%=200+RND(880)
140   Y%=200+RND(640)
150   R%=10+RND(400)
160   GCOL RND(4)-1,RND(8)-1
170   PROCfastcirc(X%,Y%,R%,TRUE)
180   UNTIL INKEY$(0)=" "
190 END
```

Enter that short program along with PROCfastcirc. When you run it, it draws randomly-positioned discs of random size, in random colours. The process continues until you press the space bar.

This technique of rapidly calculating successive trig values for steadily increasing angles can be used in all sorts of calculations. Although it is most useful for graphics, there are lots of other applications in which it can help.

You might like to think how to extend PROCfastcirc to draw ellipses. A circle is just a special form of ellipse. For starters, try an ellipse with major axis (the long one) length "2a" and minor axis "2b", centred at (centrex,centrey). Its equations are:

```
x=centrex+(a*COS(ang))
y=centrey+(b*SIN(ang))
```

When you've done that, how about drawing ellipses at angles? It's not that difficult.

Let me know of any novel applications for rapid sine and cosine calculations and I'll try to get the Editor to publish the best. Next month, I'll show you how to speed-up the calculation of maths functions.

EQU FUNCTIONS FOR BASIC I

Basic II adds some most useful directives for machine code programmers in the form of EQUS, EQUB etc. Stephen Todd has written a standard set of equivalent functions to do the same job in Basic I.

Basic II provides a most useful set of directives for machine code programmers in the form of EQUB, EQUS etc. These allow data of various types to be directly assigned to memory locations.

Unfortunately, Basic I users do not have these functions available and must resort to a more cumbersome approach, leaving the assembler, poking the relevant data into memory via the program counter (P%), incrementing the program counter and returning to the assembler.

On this page we list four Basic functions that can be used in Basic I to replace the EQU directives whenever they appear in a listing. These functions (or just those you need) should be appended to any program to be converted, and all occurrences of EQUB, EQUS, EQUW and EQUD replaced by OPT FNEQUB, FNEQUS,... as appropriate. For example:

```
EQUS "BEEBUG"
```

in a program would be replaced by

```
OPT FNEQUS("BEEBUG")
```

You will probably find it most convenient to save these functions as a spooled file. Thus type in the functions as listed and then enter:

```
*SPOOL EQUATE
LIST
*SPOOL
```

The functions can then be appended to any program by typing *EXEC EQUATE. Do make sure that the line numbers do not overlap with any in your program or some lines will be lost. Note also that the functions depend on PASS% being used to control the assembler passes.

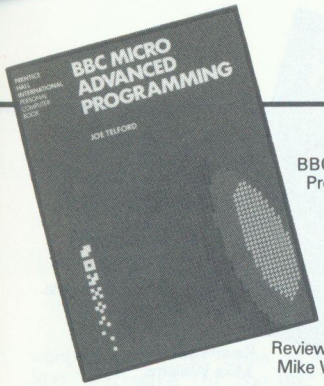
To use the other functions provided use OPT FNEQUB, OPT FNEQUW and OPT FNEQUD. When using EQUW and EQUD, the bytes of data are in reverse. So to poke the four bytes of data &12345678 into memory, use

```
OPT FNEQUD(&78563412)
```

For more details on the EQUate functions see the Advanced User Guide page 26.

The main reason for the length of the functions is that they produce an 'assembler style' output when they are called, printing out the program counter, function and relevant bytes.

```
32000 REM Set PASS% equal to pass variab
le in assembly language loop
32002 DEFFNEQUB(Byte):LOCAL byte,I$:byte
=Byte
32003 ?P%=byte:IFPASS%<>3 P%=P%+1:=PASS%
32004 IFP%<&1000PRINT;"Ø";~P%; ELSE PRIN
T;~P%;
32005 I$="":IF?P%<&10I$="Ø"
32006 I$=" "+I$+STR$~?P%:PRINTI$:P%=P%+1
:=PASS%
32008 DEFFNEQUW(Word):LOCAL word,I$,I1$:
word=Word
32009 ?P%=word MOD 256:P%?1=word DIV 256
:IFPASS%<>3 P%=P%+2:=PASS%
32010 IFP%<&1000PRINT;"Ø";~P%; ELSE PRIN
T;~P%;
32011 I$="":IF?P%<&10I$="Ø"
32012 I1$="":IF?(P%+1)<&10I1$="Ø"
32013 I$=" "+I$+STR$~?P%:I1$=" "+I1$+STR
$~?(P%+1)
32014 PRINTI$;I1$:P%=P%+2:=PASS%
32016 DEFFNEQUD(Dword):LOCAL dword,I%,I$
,I1$:dword=Dword
32017 !P%=dword:IFPASS%<>3P%=P%+4:=PASS%
32018 IFP%<&1000PRINT;"Ø";~P%; ELSE PRIN
T;~P%;
32019 I1$="":FORI%=P%TOP%+2:I$="":IF?I%<
&10I$="Ø"
32020 I1$=I1$+" "+I$+STR$~?I%:NEXT:PRINT
I1$'"    ";
32021 I$="":IF?(P%+3)<&10I$="Ø"
32022 I$=" "+I$+STR$~?(P%+3):PRINTI$:P%=
P%+4:=PASS%
32024 DEFFNEQUS(Str$):LOCAL str$,Len,C%,
I%,I$:str$=Str$:Len=LENstr$
32025 FORC%=1TOLen:?(P%+C%-1)=ASCMID$(st
r$,C%,1):NEXT
32026 C%=Ø:IFPASS%<>3 P%=P%+Len:=PASS%
32027 IFP%<&1000PRINT;"Ø";~P%; ELSE PRIN
T;~P%;
32028 I%=P%:REPEAT:I$="":IF?I%<&10I$="Ø"
32029 I$=" "+I$+STR$~?I%:PRINTI$;:I%=I%+
32030 C%=C%+1:IFC%MOD3=ØANDI%<>P%+Len C%
=Ø:PRINT'"    ";
32031 UNTILI%=P%+Len:PRINT:P%=P%+Len:=PASS%
```



BBC Micro Advanced Programming

by Joe Telford

Published by Prentice-Hall International at £9.95.

Reviewed by Mike Williams.

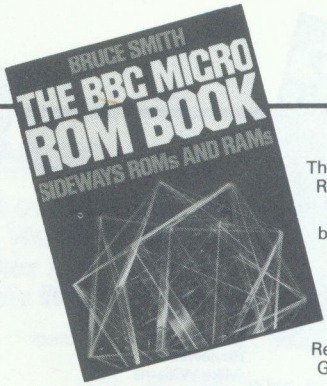
You may well question the value of yet another book on programming for the Beeb, particularly one concerned with Basic. Don't be misled. This is one of those rare books that will really help you to become a good programmer.

The book assumes that you have already learnt Basic as a computer language and know how to write, debug and run programs. What this book will do is to show you a wealth of ideas and techniques which are fundamental to the best of programming.

The author starts by discussing program structure, the use of conditions and loops, and the all important procedure and function. This is followed by a lucid and detailed look at data structures including lists, stacks, queues, linked lists and tree structures. Following chapters deal with input/output, sorting, file handling, logic, binary, etc, etc. The emphasis throughout is on using your knowledge of Basic to build an understanding of a variety of techniques fundamental to the best of programming practice.

Later chapters appear more conventional in content, covering the almost de rigeur use of graphics, sound and elementary machine code. There is also a useful chapter on programming the various peripheral ports.

If you are familiar with Basic and want to develop your programming skills further then I strongly recommend that you consider this book. It contains much that I applaud, but some of the later chapters might with benefit have been omitted (and the price brought down as well).



The BBC micro ROM Book

by Bruce Smith,

published by Collins for £9.95.

Reviewed by Geoff Bains.

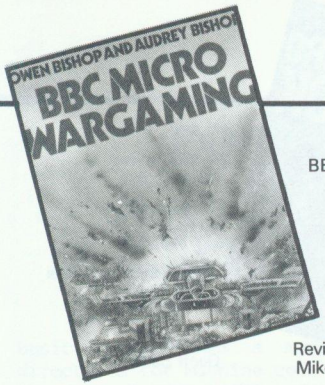
This is such a good idea for a book that it is amazing nobody has done it before. The ROM book covers just about every aspect of sideways ROMs on the Beeb, though it has to said that there is a fair amount of what could be described as padding in this 280 page tome.

The first half of the book is definitely not padded. It kicks off with a description of what ROMs and EPROMs are and how the sideways ROM (or RAM) system is implemented. A very detailed chapter on each of service and language ROMs is given, with another on the ROM filing system. Emphasis here is on writing your own ROM and several example routines to handle the protocols are given.

Further chapters give a complete disassembly listing of BEEBUGSOFT's Toolkit command interpreter and Addcomm's BRK error trapping routine. Another takes you through a worked example utility ROM. To round it all off there are details of the construction (in only moderate depth) of an EPROM (8K only) programmer and the software to run it.

The latter half of the book contains reviews of various ROMs (including BEEBUGSOFT's Toolkit) and ROM boards. Then comes the padding, 70 pages of barcode listings of the programs (have YOU got a barcode reader?). Despite these gripes this book is still excellent. It covers everything you always wanted to know about ROMs (but, of course were afraid to ask). This is a heavy subject at the best of times but the smatterings of characteristic Smith humour make it both informal and informative. Definitely recommended.

BOOK REVIEWS



BBC Micro Wargaming

by Owen & Audrey Bishop

Published by Collins at £8.95.

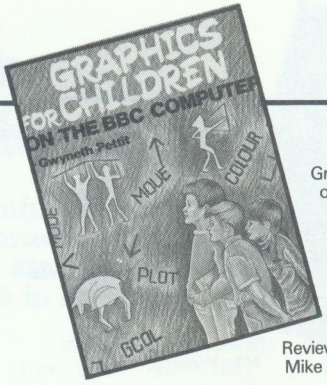
Reviewed by Mike Williams

The authors claim that their book is designed both for the programmer with an interest in war games, and for the experienced wargamer seeking to use the BBC micro for this purpose. In my view the programmer is the clear winner in this particular skirmish. The wargamer will need considerable determination to achieve the same target. It is also a shame that the wargame programs listed in the book are not available on cassette or disc.

Following initial briefing on the use of tables and the storage of data we are introduced to our first wargame, based perhaps inevitably on a World War II scenario. Much ammunition in the form of numerical data is needed to build the model on which the game program acts. It might at least have been worthwhile to have included some simple form of error checking here. That apart, all is clearly and comprehensively described. I felt that most readers would have welcomed more discussion on the 'how' and the 'why' of the game, and how the ideas of wargaming are implemented within the program.

A number of utilities for the benefit of the programming wargamer are described before an enhanced version of the first wargame (more data, longer program). Three other wargames are presented for inspection, the Norman invasion of 1066, the siege of Napoleon in Paris in 1814 and BENA 2352, a wargame of the future.

Overall, this book of 230 pages contains a heavy onslaught of information that the mere casual reader may well find overwhelming. For the more resolute, there is much to be gained from this book.



Graphics for Children on the BBC Computer

by Gwyneth Pettit

Published by Sigma Technical Press at £6.95

Reviewed by Mike Williams

I had expected more of this book. Graphics is certainly one of the most exciting and stimulating aspects of many micros including the Beeb, that it should be possible to inspire many children in this direction with a dazzling compendium of example and technique. In contrast, the content of this book is very conventional, very much a text book in style with exercises at the end of each section. And there are so few illustrations!

Overcoming the initial disappointment, the book is well written, clearly with children in mind, but without being at all patronising. Indeed, many adults new to the BBC micro might well find this book a most useful introduction to graphics. All the basic techniques are well described, though despite a whole chapter to itself, I would wish more attention had been given to animation, admittedly not always an easy topic, but one which is usually basic to the many commercial computer games that children play. There is also very little devoted to user-defined characters.

Apart from the general chapters on graphics (the heart of the book) there are also three more applications oriented ones covering business graphics, mathematical graphics and three-dimensional modelling. There is also a short chapter devoted to various graphics input and output devices, including three short joystick routines.

There is much to commend this book, and yet I feel it could have achieved so much more. It is a workmanlike book ideal for many a school library. I am not sure that it will have the spontaneous appeal to the one group of people it should - children!

BOOK REVIEWS

Software Compatibility on the B+

Following the initial report on the new BBC model B+ in the June BEEBUG, Alan Webster checks out the problems you may encounter if you try to run all your existing software on the new Beeb.

The following table shows the compatibility of some of the most popular ROM software for the BBC micro on the new model B+.

The main area of incompatibility is in the use of shadow screen RAM and modes 128 to 135. These use a different area of memory for the display, and subsequently, most ROMs that directly access screen memory will not work properly.

Most of the ROMs tested below still work on the B+, although some cannot cope with shadow memory. Only one of the ROMs tested worked both on the B+ and made use of the extra memory available, and that was Viewsheet.

Product	Company	Compatibility	Product	Company	Compatibility
Accelerator	CC	****	Toolkit	BB	****
Beebed	JO	***	Ultracalc	BS	****
Disc Doctor	CC	***	Vasm	VR	****
Exmon 2	BB	***	View 1.4	AC	**
Graphics Rom	CC	****	View 2.1	AC	**
Help	BB	****	Viewsheet	AC	****
Murom	BB	*	Wordwise	CC	***
Printmaster	CC	****	Wordwise B20	CC	***
Sleuth	BB	***	Wordwise Plus	CC	***
Spellcheck	BB	****			

Key:

**** - Works in both normal and shadow modes.
*** - Only parts work in shadow modes.
** - Doesn't work fully in shadow or normal modes.
* - Doesn't work at all.

Companies:

AC Acornsoft
BB Beebugsoft
BS BBC Soft
CC Computer Concepts
JO J and O Software
VR Vida Rebus

HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TORCH RESET - Wim Bailleul

The Torch Z80 second processor sets PAGE to &2800. When using the machine as a Beeb this can be reset to &1900 with PAGE but it is quicker to press Break and hold down B.

DELETING WORDWISE TEXT - Jonathon Evans

Whereas pressing Break no longer erases text in Wordwise Plus, you can start afresh by typing :NEW from a menu. Alternatively, either the main text or any one segment can be deleted by typing :DELETE TEXT from the menu of the segment concerned.

INTERRUPTING ARIES - J.W.E.van Dijk

When using interrupts or events with the Aries B-20 board you should locate the interrupt routine below &3000, otherwise the event and interrupt handling routines in the OS are liable to look in the wrong RAM for your code.

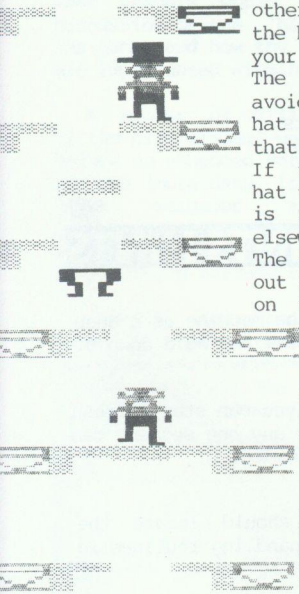
Hatrace

Richard Lewis, who wrote the intriguing Spider Man game published in the March BEEBUG, describes his latest action-packed game, Hatrace. Nimble fingers and fast reactions are called for in this two-player game.

Hat Race is an entertaining two player game which takes place in a hat factory. The object of the game is to collect the hats that have been left lying about on the shop floor. The winner of the game is the one who collects the greater share of the twenty one hats.

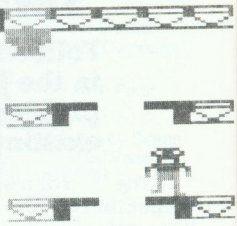
The factory layout displayed on the screen consists of many platforms and girders, linked by elevators. Each player starts at the top of the screen and must make his or her own way around the factory to collect a hat. Once a player has collected a hat, it must then be taken back to their own store at the top of the screen to score a point.

When a player is wearing a hat, there are three possible hazards or dangers. The other player may steal the hat from right under your nose (or above it!). The second problem is to avoid getting ink on the hat from the ink pots that fall from the roof. If this happens then the hat that you are carrying is randomly positioned elsewhere on the screen. The third thing to watch out for are the ink spots on the floor (coloured magenta). If you tread on one of these, you will again lose your hat.

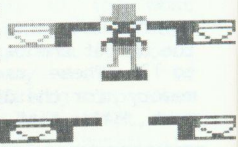


The first stage contains one hat to be collected before you are allowed a breather.

The second stage will then start and this will have two hats to collect. This continues until the sixth stage when six hats have to be collected. Once all six hats have been stored then the game ends.



The keys to control the two 'people' in this game are 'Z' and 'X' to move player one left and right, and '>' and '?' do likewise for player two.



Hat Race is written entirely in Basic, yet it is fast, colourful and smooth.



PROGRAM NOTES

The program is quite large, and so has to run in mode 5. There are several small procedures that are called from within the main procedures, and it is these main procedures that are outlined below.

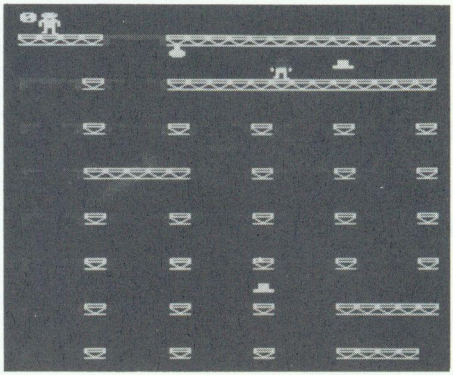
Procedure	Description
PROCMA	Set up and draw playing screen
PROCGM	Place hats randomly on screen
PROCCHAR	Define characters
PROCS	Define print strings
PROCman	Move and test players
PROCSCR	Print score
PROCG	Set up and print control screen

The strings in lines 1970 to 2130 are defined in this way to make the printing of characters on the screen much faster. If, for example, you have already run the program, with the strings still set up, then printing M$(1) or M$(2) will quickly display one of the men on the screen.

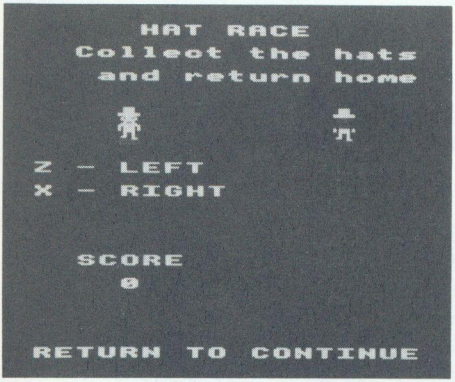
```
  10 REM Program Hat Race
  20 REM Version B0.3
  30 REM Author  R. Lewis
  40 REM BEEBUG  July 1985
  50 REM Program subject to copyright
  60 :
 100 ON ERROR GOTO 200
 110 MODE5:PROCCHAR
 120 DIM MX%(1),MY%(1),MZ%(1),ML%(1),HT
%(1),M%(1),SC%(1)
```

```
  130 DIMX%(6),Y%(6),U%(6),L%(6),D%(6),I
%(6),U$(6),D$(6),MU$(1),MD$(1),L$(3),R$(
3),M$(3),A%330
  140 PROCS:NM%=0:HSC%=0:hsc%=0:SC%(0)=0
:SC%(1)=0
  150 VDU19,3,5,0,0,0
  160 REPEAT:PROCRM:PROCG:PROCMA:PROCSCR
  170 REPEAT:PROClift:Q%=0:PROCman:PROCi
nk:Q%=1:PROCman:UNTILOV%
  180 UNTIL FALSE
  190 :
  200 ON ERROR OFF
  210 MODE7:IF ERR=17 END
  220 REPORT:PRINT " at line ";ERL
  230 END
  240 :
 1000 DEFPROCMA
 1010 CLS:CF%=FALSE:VDU23,1,0;0;0;0;
 1020 PROCGM:COLOUR2
 1030 FORI%=3TO27STEP4:PRINTTAB(0,I%)STR
ING$(20,CHR$240);:NEXT:PRINTTAB(0,31)STR
ING$(19,CHR$240);
 1040 FORI%=A%TOA%+287STEP21:$I%=STRING$
(20,CHR$7)+CHR$0:NEXT
 1050 A%?166=0:COLOUR1
 1060 FORI%=1TO6:FORJ%=U%(I%)TOL%(I%)
 1070 IF (J%+1) MOD 4=0 COLOUR1:PRINTTAB
(X%(I%)-1,J%)CHR$252LB$CHR$253;:COLOUR1
ELSEPRINTTAB(X%(I%),J%)LB$;
 1080 K%=((J%+1) DIV 4-1)*21+A%+X%(I%)
 1090 ?K%=I%:NEXT:I%(I%)=0
 1100 PRINTTAB(X%(I%),Y%(I%))L$;:NEXT:I%
(0)=0
 1110 FORI%=1TONM%:PROCSM:NEXT
 1120 ?A%=0:A%?19=0
 1130 ENDPROC
 1140 :
 1150 DEFPROCSA
 1160 J%=RND(18):K%=RND(7):H%=A%+J%+K%*2
1
 1170 IF?H%<>7OR?(H%-21)<>7OR?(H%+1)<>7O
R?(H%-1)<>7OR?(H%-20)<>7OR?(H%-22)<>7GOT
O1160
 1180 ENDPROC
 1190 :
 1200 DEFPROCSM:PROCSA:?H%=9:COLOUR3:PRI
NTTAB(J%,K%*4+1)CHR$242;:ENDPROC
 1210 :
 1220 DEFPROCRM
 1230 MX%(0)=1:MX%(1)=18:MY%(0)=1:MY%(1)
=1:M%(0)=1:M%(1)=3:MZ%(0)=A%:MZ%(1)=A%:H
T%(0)=0:HT%(1)=0:ML%(0)=0:ML%(1)=0:OV%=F
ALSE
 1240 AX%=0:AY%=1:OAX%=0:OAY%=1
 1250 ENDPROC
 1260 :
 1270 DEFPROCGM
 1280 X%(1)=FNRX:U%(1)=3:L%(1)=U%(1)+4+R
ND(4)*4
 1290 X%(2)=FNRX:IFX%(2)=X%(1)GOTO1290
 1300 L%(2)=31:U%(2)=L%(1)-RND(2)*4
 1310 FORI%=3TO6
 1320 X%(I%)=FNRX:K%=0
 1330 FORJ%=1TOI%-1
 1340 IFX%(I%)<>X%(J%)GOTO1360
 1350 IFK%=0K%=J%ELSEK%=-1
 1360 NEXT
 1370 IFK%=0PROCNL:GOTO1420
 1380 IFK%=-1GOTO1320
 1390 U%(I%)=U%(K%)-3:L%(I%)=31-L%(K%)
 1400 IFU%(I%)<12ANDL%(I%)<12GOTO1320
 1410 IFU%(I%)<L%(I%)PROCNB ELSE PROCNT
 1420 NEXT
 1430 cro%=FALSE:FORI%=1TO6:IF U%(I%)=3
AND (X%(I%)=1 OR X%(I%)=17) THEN cro%=TR
UE
 1440 NEXT:IF cro%=TRUE GOTO 1280
 1450 FORI%=1TO6:U$(I%)=LU$:D$(I%)=LD$:D
%(I%)=RND(2)-1:Y%(I%)=(U%(I%)+L%(I%))/2:
NEXT
 1460 ENDPROC
 1470 :
 1480 DEFPROCNL
 1490 U%(I%)=-1+RND(5)*4:L%(I%)=U%(I%)+8
+4*RND(3):IFL%(I%)>31L%(I%)=31
 1500 ENDPROC
 1510 :
 1520 DEFPROCNT
 1530 U%(I%)=-1+RND(2)*4:L%(I%)=U%(I%)+4
+RND(2)*4
 1540 IFL%(I%)>=U%(K%)GOTO1530
 1550 ENDPROC
 1560 :
 1570 DEFPROCNB
 1580 L%(I%)=35-RND(2)*4:U%(I%)=L%(I%)-4
-4*RND(2)
 1590 IFU%(I%)<=L%(K%)GOTO1580
 1600 ENDPROC
 1610 :
 1620 DEFFNRX=4*RND(5)-3
 1630 :
 1640 DEFPROClift
 1650 COLOUR1
 1660 FORI%=1TO6
 1670 IFD%(I%)GOTO1700
 1680 PRINTTAB(X%(I%),Y%(I%))U$(I%);:Y%(
I%)=Y%(I%)-2:IFY%(I%)=U%(I%)D%(I%)=TRUE
 1690 NEXT:ENDPROC
 1700 PRINTTAB(X%(I%),Y%(I%))D$(I%);:Y%(
I%)=Y%(I%)+2:IFY%(I%)=L%(I%)D%(I%)=FALSE
 1710 NEXT:ENDPROC
 1720 :
 1730 DEFPROCCHAR
 1740 VDU23,236,255,195,195,195,102,36,3
6,102
 1750 VDU23,237,255,255,255,255,126,60,6
0,126
 1760 VDU23,238,24,24,24,60,126,126,126,
60
```

```
 1770 VDU23,239,24,24,24,60,126,126,126,
60
 1780 VDU23,240,255,129,255,129,66,36,24
,255
 1790 VDU23,241,255,255,255,255,0,0,0,0
 1800 VDU23,242,0,0,60,60,60,60,255,255
 1810 VDU23,243,60,90,126,60,36,126,255,
189
 1820 VDU23,244,189,189,60,36,36,36,100,
6
 1830 VDU23,245,189,189,60,36,36,36,38,9
6
 1840 VDU23,246,0,0,0,0,0,0,0,0
 1850 VDU23,247,0,0,0,0,0,0,0,0
 1860 VDU23,252,255,255,255,224,224,224,
224,224
 1870 VDU23,253,255,255,255,7,7,7,7,7
 1880 VDU23,224,28,48,62,20,28,60,126,18
9
 1890 VDU23,225,189,60,24,24,56,104,200,
76
 1900 VDU23,226,189,60,24,24,28,23,18,24
 1910 VDU23,227,189,60,24,24,28,22,19,50
 1920 VDU23,228,189,60,24,24,56,232,72,2
4
 1930 VDU23,229,28,6,62,20,28,60,126,189
 1940 ENVELOPE1,1,6,0,-6,200,100,200,100
,2,0,-1,120,110
 1950 ENDPROC
 1960 :
 1970 DEFPROCS
 1980 A$=CHR$8:C$=CHR$10:F$=CHR$11:TE$=C
$+A$+A$
 1990 ET$=A$+C$:EE$=A$+F$
 2000 EG$=F$+A$:R$=CHR$17+CHR$1
 2010 Y$=CHR$17+CHR$2:W$=CHR$17+CHR$3
 2020 B$=F$+CHR$32+ET$+CHR$32+C$+A$+CHR$
32
 2030 L$=CHR$241:LB$=CHR$32
 2040 LU$=LB$+F$+EG$+L$
 2050 LD$=LB$+C$+C$+A$+L$
 2060 FORI%=0TO1:Q$=CHR$17+CHR$(2-I%):TP
$=LB$+EG$+LB$+EE$+L$+EE$+W$+CHR$244
 2070 MU$(I%)=TP$+Q$+EE$+CHR$243+W$+EE$+
CHR$(246+I%)+R$
 2080 TP$=LB$+EE$+F$+LB$+EE$+LB$+C$+ET$+
LB$+ET$+CHR$243
 2090 MD$(I%)=Q$+TP$+EE$+W$+CHR$(246+I%)
+C$+ET$+W$+CHR$244+ET$+R$+L$
 2100 TP$=A$+CHR$229+LB$+TE$+W$:L$(2*I%)
=W$+EE$+CHR$(246+I%)+LB$+ET$+Q$+TP$+CHR$
227+LB$:L$(1+2*I%)=W$+EE$+CHR$(246+I%)+L
B$+ET$+Q$+TP$+CHR$228+LB$
 2110 TP$=LB$+CHR$224+TE$+LB$+W$:R$(2*I%
)=F$+LB$+W$+CHR$(246+I%)+TE$+Q$+TP$+CHR$
225:R$(1+2*I%)=F$+LB$+W$+CHR$(246+I%)+TE
$+Q$+TP$+CHR$226
 2120 TP$=CHR$243+C$+A$+W$:M$(2*I%)=F$+W
$+CHR$(246+I%)+ET$+Q$+TP$+CHR$244:M$(1+2
```



```
*I%)=F$+W$+CHR$(246+I%)+ET$+Q$+TP$+CHR$2
45
 2130 M$(2)=F$+W$+CHR$247+ET$+R$+TP$+CHR
$244:M$(3)=F$+W$+CHR$247+ET$+R$+TP$+CHR$
245
 2140 NEXT:?A%=0:A%=A%+1
 2150 ENDPROC
 2160 :
 2170 DEFPROCman
 2180 P%=MZ%(Q%)+MX%(Q%):IF?P%<7GOTO2320
 2190 M%(Q%)=1-M%(Q%)+4*Q%:M%=M%(Q%)
 2200 IF?P%=9 AND HT%(Q%)=0 THEN HT%(Q%)
=1:VDU23,246+Q%,0,0,60,60,60,60,255,255:
?P%=7:A%?(1+17*Q%)=10+Q%:COLOUR3:PRINTTA
B(MX%(Q%),MY%(Q%)-1)CHR$(246+Q%):SOUND1,
1,200,20:ENDPROC
 2210 IF?P%=10+Q% THEN ?P%=7:SC%(Q%)=SC%
(Q%)+HT%(Q%):HT%(Q%)=0:VDU23,246+Q%,0,0,
0,0,0,0,0,0:PROCSCR:SOUND1,1,200,20:ENDP
ROC
 2220 IF ?P%=30 THEN COLOUR2:PRINTTAB(MX
%(Q%),MY%(Q%)+2);CHR$240;:?P%=7 ELSE GOT
O 2240
 2230 IF HT%(Q%)=1 THEN PROCOFF(Q%):PROC
SM ELSE SOUND1,1,200,6
 2240 IFINKEY-(67+38*Q%)GOTO2260ELSE IF
INKEY-(98+6*Q%)GOTO2290
 2250 PRINTTAB(MX%(Q%),MY%(Q%))M$(M%):EN
DPROC
 2260 PROCTEST(1):IFP%?1>6PRINTTAB(MX%(Q
%),MY%(Q%))R$(M%):MX%(Q%)=MX%(Q%)+1ELSE
ML%(Q%)=P%?1:PROCEN
 2270 IF?P%=9COLOUR3:PRINTTAB(MX%(Q%)-1,
MY%(Q%))CHR$242
 2280 ENDPROC
 2290 PROCTEST(0):IFP%?-1>6PRINTTAB(MX%(
Q%),MY%(Q%))L$(M%):MX%(Q%)=MX%(Q%)-1ELSE
 ML%(Q%)=P%?-1:PROCEN
 2300 IF?P%=9COLOUR3:PRINTTAB(MX%(Q%)+1,
MY%(Q%))CHR$242
 2310 ENDPROC
```




```
 2320 IFINKEY-(67+38*Q%) TX%=1:PROCEX:EN
DPROC
 2330 IFINKEY-(98+6*Q%) TX%=-1:PROCEX:EN
DPROC
 2340 MY%(Q%)=Y%(ML%(Q%))
 2350 ENDPROC
 2360 :
 2370 DEFPROCEN
 2380 IFML%(Q%)=0GOTO2450
 2390 IFY%(ML%(Q%))<>MY%(Q%)+20RI%(ML%(Q
%))PRINTTAB(MX%(Q%),MY%(Q%))M$(M%):ENDPR
OC
 2400 SOUND1,1,100,2:PRINTTAB(MX%(Q%),MY
%(Q%))B$
 2410 MX%(Q%)=X%(ML%(Q%)):U$(ML%(Q%))=MU
$(Q%):D$(ML%(Q%))=MD$(Q%)
 2420 PRINTTAB(MX%(Q%),MY%(Q%))M$(M%)
 2430 I%(ML%(Q%))=9
 2440 ENDPROC
 2450 IFMX%(Q%)=0PRINTTAB(0,MY%(Q%))B$:M
X%(Q%)=19+(MY%(Q%)=29):PRINTTAB(MX%(Q%),
MY%(Q%))M$(M%):ENDPROC
 2460 IFMX%(Q%)>18 OR P%=A%+165 PRINTTAB
(MX%(Q%),MY%(Q%))B$:MX%(Q%)=0:PRINTTAB(M
X%(Q%),MY%(Q%))M$(M%):ENDPROC
 2470 PRINTTAB(MX%(Q%),MY%(Q%))M$(M%)
 2480 ENDPROC
 2490 :
 2500 DEFPROCEX
 2510 IF(Y%(ML%(Q%))+1) MOD4=0 ELSE ENDP
ROC
 2520 SOUND1,1,200,2
 2530 U$(ML%(Q%))=LU$:D$(ML%(Q%))=LD$
 2540 I%(ML%(Q%))=0
 2550 MY%(Q%)=Y%(ML%(Q%))-2:MZ%(Q%)=((Y%
(ML%(Q%))+1) DIV4-1)*21+A%
 2560 PRINTTAB(MX%(Q%),MY%(Q%))B$:MX%(Q%
)=MX%(Q%)+TX%:ML%(Q%)=0
 2570 PRINTTAB(MX%(Q%),MY%(Q%))M$(M%(Q%)
):ENDPROC
 2580 :
 2590 DEFPROCSCR
 2600 COLOUR2:PRINTTAB(0,1);SC%(0);:COLO
UR1:PRINTTAB(19,1);SC%(1);
 2610 IF SC%(0)+SC%(1)=NM% THEN OV%=TRUE
 2620 ENDPROC
 2630 :
 2640 DEFPROCTEST(BB%)
 2650 IF MX%(Q%)<>(MX%(1-Q%)+1+18*BB%)MO
D 20 ORMY%(Q%)<>MY%(1-Q%) THEN ENDPROC
 2660 IF HT%(Q%)=0 AND HT%(1-Q%)=1 THEN
HT%(Q%)=1:VDU23,246+Q%,0,0,60,60,60,60,2
55,255:A%?(1+17*Q%)=10+Q%:PROCOFF(1-Q%)
 2670 ENDPROC
 2680 DEFPROCG
 2690 HSC%=HSC%+SC%(0):hsc%=hsc%+SC%(1):
SC%(0)=0:SC%(1)=0
 2700 CLS:COLOUR3:PRINTTAB(5,2)"HAT RACE
";:PRINTTAB(2,4)"Collect the hats";:PRIN
TTAB(2,6)" and return home";
 2710 PRINTTAB(4,10)MD$(0):COLOUR2:PRINT
TAB(0,14)"Z - LEFT";:PRINTTAB(0,16)"X -
RIGHT";:PRINTTAB(2,22)"SCORE";:PRINTTAB(
4,24);HSC%;
 2720 PRINTTAB(14,10)MD$(1):COLOUR1:PRIN
TTAB(10,14)". - LEFT";:PRINTTAB(10,16)"/
 - RIGHT";:PRINTTAB(12,22)"SCORE";:PRINT
TAB(14,24);hsc%;
 2730 COLOUR3:PRINTTAB(4,9)CHR$242;:PRIN
TTAB(14,9)CHR$242;
 2740 NM%=NM%+1:IF NM%<7 THEN 2770
 2750 NM%=1:COLOUR3:PRINTTAB(5,17)"GAME
OVER":IF HSC%>hsc% THEN COLOUR2:PRINTTAB
(2,19)"I WIN"ELSE COLOUR1:PRINTTAB(12,19
)"I WIN"
 2760 HSC%=0:hsc%=0
 2770 COLOUR3:PRINTTAB(0,30)"RETURN TO C
ONTINUE";
 2780 REPEAT UNTIL INKEY-74
 2790 ENDPROC
 2800 :
 2810 DEFPROCink
 2820 AX%=AX%-1:IF AX%>=0 GOTO2850
 2830 AX%=19:AY%=RND(7):AD%=4*RND(4)-1:A
U%=3
 2840 AP%=A%+AD%+AY%*21:IF ?AP%<>7 AD%=-
1:AU%=0
 2850 COLOUR3:PRINTTAB(AX%,4*AY%);CHR$(2
37+(AX%<AD%)-(AX%=AD%));
 2860 PRINTTAB(OAX%,4*OAY%);" ";
 2870 IF AX%<AD% AND AU%>0 THEN PRINTTAB
(AD%,4*AY%-AU%+4);CHR$(241-AU%);:AU%=AU%
-1:IF AU%=1 OR AU%=0 THEN PRINTTAB(AD%,4
*AY%-AU%+2);" ";:?AP%=30+AU%
 2880 OAX%=AX%:OAY%=AY%:FORI%=0TO1
 2890 IF (AX%=MX%(I%) OR AX%=MX%(I%)+1)
AND AY%*4=MY%(I%)-1 AND HT%(I%)=1 THEN P
ROCOFF(I%):PROCSM
 2900 NEXT:ENDPROC
 2910 :
 2920 DEFPROCOFF(CC%):HT%(CC%)=0:VDU23,2
46+CC%,0,0,0,0,0,0,0,0:A%?(1+17*CC%)=7:S
OUND1,1,200,20:ENDPROC
```

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams
Assistant Editor: Geoff Bains
Production Editor: Phyllida Vanstone
Technical Assistant: Alan Webster
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft
Additional thanks are due to Sheridan Williams, Adrian Calcraft, John Yale and Tim Powys-Lybbe.

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility, whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Publications Limited.
BEEBUG Publications Ltd (c) 1985

Editorial Address
BEEBUG
PO BOX 50
St. Albans
Herts.

CONTRIBUTING TO BEEBUG

PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £40 per page, but please give us warning of anything substantial that you intend to write. A leaflet, 'Notes of Guidance for Contributors' is available on receipt of an A5 (or larger) SAE.

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", "View", or other means, but please ensure an adequate written description of your contribution is also included. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month, plus one of £15 for a hint or tip deemed to be exceptionally good.

Please send all editorial material to the editorial address below. If you require a reply it is essential to quote your membership number and enclose an SAE.

SUBSCRIPTIONS

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

MEMBERSHIP SUBSCRIPTION RATES

£ 6.40 6 months (5 issues) UK ONLY
£11.90 UK - 1 year (10 issues)
£18 Europe, £21 Middle East
£23 Americas & Africa, £25 Elsewhere

BACK ISSUES
(Members only)

Vol.	Single Issues	Volume sets (10 issues)
1	80p	£7
2	90p	£8
3	£1	£9
4	£1	–

Please add the cost of post and packing as shown:

DESTINATION	First issue	Each subsequent issue
UK	30p	10p
Europe	70p	20p
Elsewhere	£1.50	50p

All overseas items are sent airmail (please send a sterling cheque). We will accept official UK orders but please note that there will be a £1 handling charge for orders under £10 that require an invoice. Note that there is no VAT on magazines.

Back issues are for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

Subscriptions, Back Issues &
Software Address

BEEBUG
PO BOX 109
High Wycombe
Bucks. HP10 8HQ

Hotline for queries and software orders

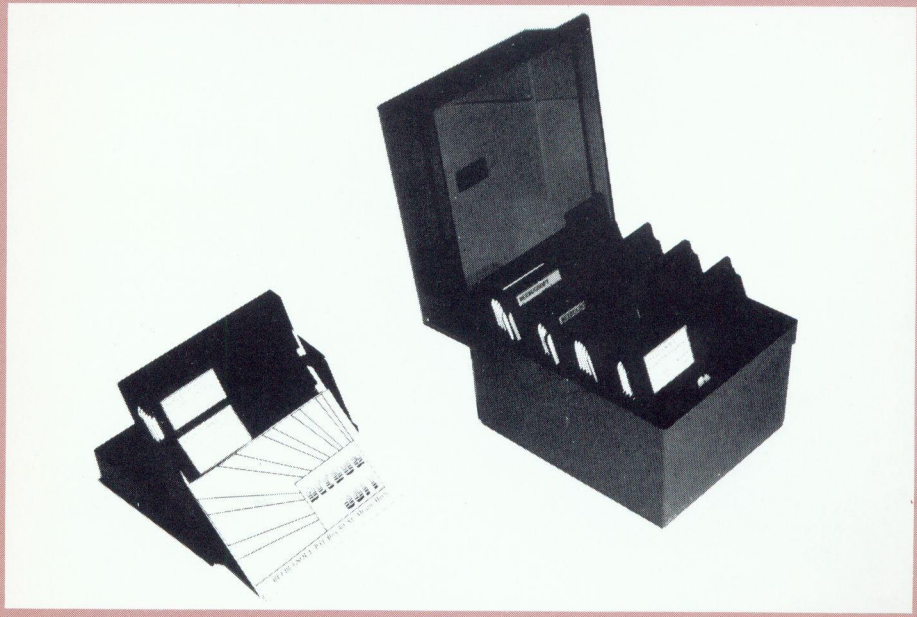
St. Albans (0727) 60263
Manned Mon-Fri 9am-4.30pm

24hr Answerphone Service for Access and Barclaycard orders, and subscriptions
Penn (049481) 6666

If you require members' discount on software it is essential to quote your membership number and claim the discount when ordering.

High Quality Low Priced Discs

Backed by The Reputation of BEEBUG



10 S/S D/D Discs – £13.90	10 D/S D/D Discs – £19.40
25 S/S D/D Discs – £33.45	25 D/S D/D Discs – £46.95
50 S/S D/D Discs – £59.30	50 D/S D/D Discs – £87.05

All Prices Include Storage Box, VAT and Delivery to Your Home (UK).

All discs are 100% individually tested, supplied with hub ring as standard, and guaranteed error free. They are ideal for use on the BBC Micro and have performed perfectly in extensive tests at BEEBUG over many months.

Orders for 25 or 50 are delivered in strong plastic storage boxes with four dividers. Orders for 10 are sent in smaller hinged plastic library cases.

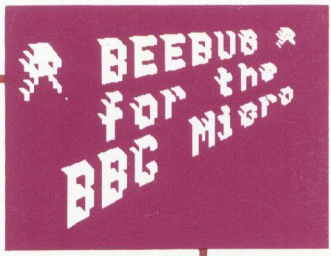
We are also able to offer the empty storage container, which holds up to 50 discs for £10 including VAT and post.

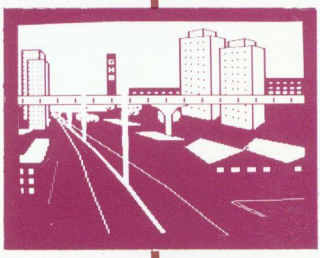
Please use the order form enclosed
or order directly from:
BEEBUGSOFT, P.O. Box 109,
High Wycombe, Bucks HP10 8HQ.

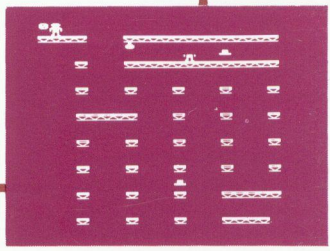
BEEBUG SOFT

Magazine Cassette/Disc

JULY 1985 CASSETTE DISC CONTENTS







3D TEXT in six different styles

JIGSAW – to turn your graphics displays into intriguing puzzles

VIEW PRINTER DRIVER GENERATOR – for any printer

SINGLE DRIVE DISC BACK-UP to make life easy

WORKSHOP PROCEDURES – fast circle drawing

MODE 7 GRAPHICS – Move, Draw and Plot

EQU FUNCTIONS for Basic I users

BEGINNERS EXAMPLES – more on using logic

HATRACE – a fast moving two-player game in the hat factory

All this for £3.00 (cass) £4.75 (disc) +50p p&p.
Back issues (disc since Vol.3 No. 1, cass since Vol.1 No. 10) available at the same prices.

Subscription rates	DISC UK	CASS UK	DISC O'seas	CASS O'seas
6 months (5 issues)	£25	£17	£30	£20
12 months (10 issues)	£50	£33	£56	£39

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to disc subscriptions on receipt of £1.70 per issue of the subscription left to run.

All subscriptions and individual orders to
BEEBUGSOFT, PO BOX 109, High Wycombe, Bucks, HP10 8NP.

Printed in England by Staples Printers St Albans Limited at The Priory Press.

ISSN 0263-7561