£1.20

BEEBUG

FOR THE BBC MICRO

Vol 3 No 7 December 1984

Applications

Graphics

Utilities

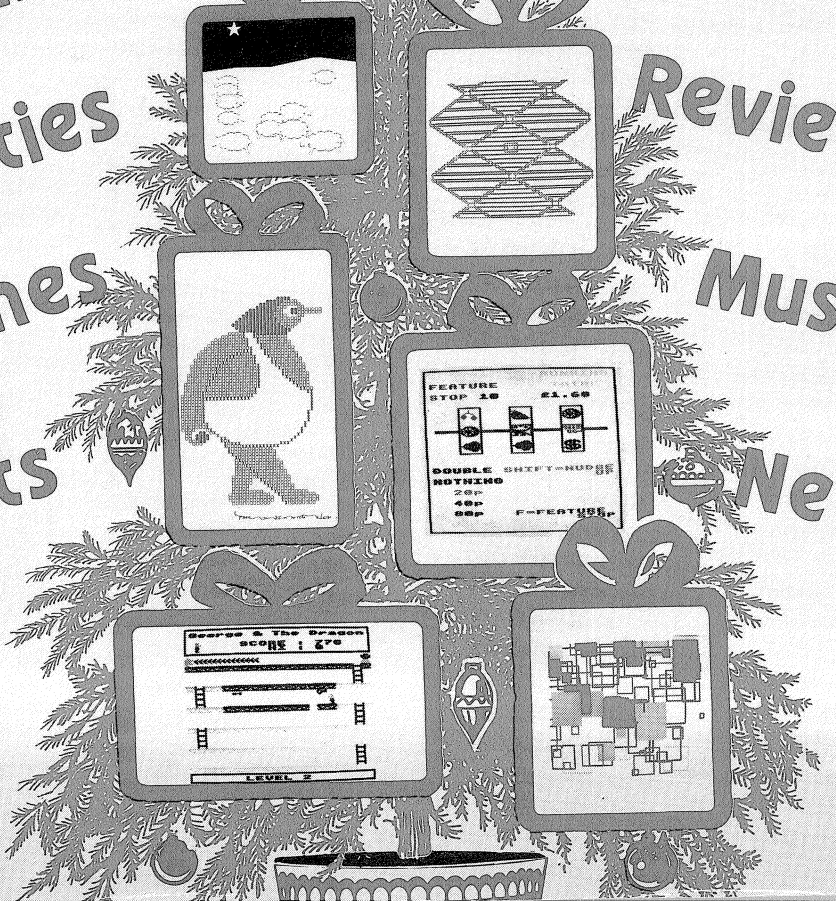
Reviews

Games

Music

Hints

News



EDITORIAL

CHRISTMAS ISSUE

This is the Christmas issue of BEEBUG and for the first time boasts over 50 pages. What is more, future issues will also provide just as many pages, with more reviews and news, and even more programs and articles.

To provide the Christmas flavour this month, we have a musical Christmas Carol (with three more on the magazine cassette/disc), and a Cartoon Calendar for 1985 featuring a character from a well-known and popular computer game. As well, in addition to our usual two excellent full-length games, we have one page containing no less than three complete and challenging one-line games.

MEMORY SIZE

Starting from the next issue, we shall no longer publicise the relevant memory size (16K or 32K) with the title of each program in the magazine. The number of remaining 16K machines is now a small and decreasing proportion of the total, and the information can be inferred in most cases from mode statements in the programs.

We shall continue to test all our programs on both Basic I and Basic II, and we also check all programs for disc or tape incompatibilities where appropriate.

Mike Williams

NOTICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOA

TESTING OUT YOUR MICRO

We have had a number of enquiries regarding the disc drive test program published in BEEBUG Vol.3 No.5. Although this was tested with a variety of disc drives in use here at BEEBUG, it is clear that the program can produce a misleading result with some drives (or with some settings). if your drives otherwise appear to be working quite correctly, then you should ignore any indication to the contrary produced by the test program (though it may be worth checking the settings of the DIL switch under the cover at the front of the keyboard for correct timimg). In particular, if you have an older (and slower) Shugart drive, for example model SA400L, you should try changing the '-21' at line 1270 to '-16' to match the timimg.

WEE SHUGGY

Unfortunately, the superb game Wee Shuggy in the last issue (Vol.3 No.7) was marred by the omission of spaces in lines 290,300 and 1900. These should be inserted between keywords and variable names in these three lines when typing in the program. This arose because the version used had been previously compacted by the author. The compaction causes no problem on the magazine cassette/disc version.

HINT WINNERS

We have decided to award an additional prize of £15 for any really outstanding hint that we publish, in addition to our normal prizes of £10 and £5. This month the £10 prize goes to Paul Walls, and we are awarding two prizes of £5, to Ashley Denninson and Tony Walsh. If you have any useful hints or tips, then why not write and let us know. Maybe you will be the first to win our new £15 prize.

MAGAZINE CASSETTE/DISC

This month the magazine cassette/disc contains a mammoth 60K of programs. This includes the full updated printer spooler utility, a total of four visual and musical Christmas carols, and an extra program: a superb professional machine code 'Pengo'-style game with a Christmas theme called Christmas Antics, by John Wallace.

BEEBUG MAGAZINE

GENERAL CONTENTS

2 Editorial
4 A Christmas Carol
6 Book Reviews
8 Points Arising
9 A Page of Games to Play
10 Extending the PLOT Instruction
14 News
15 Games Galore Reviewed
18 Printer Spooler Updated
20 Cartoon Calendar for 1985
23 External ROM Sockets Reviewed
26 Improved Trace Facility
28 Adventure Games
29 Floppy Tape Drives Reviewed
32 BEEBUG Workshop
Using Indirection Operators (Part 1)
34 Sounding out the Beeb
Two Books Reviewed
36 Build your own Graphics Tablet (Part 2)
39 Beginners Start Here
Debugging Programs (Part 2)
42 George and the Dragon
46 Christmas Fruit Machine

PROGRAMS

4 Christmas Carol
9 Three Games
10 Extended PLOT Demonstration
18 Printer Spooler Updated
20 Cartoon Calendar
26 Improved Trace Routine
32 Two Workshop Examples
36 Basic Graphics Tablet Program
42 George and the Dragon Game
46 Fruit Machine Game

HINTS, TIPS & INFO

5 Acorn Tube Screens
5 New Way to Crash a Beeb
5 Reserving Basic II Memory
19 ULA or Semi-custom Chip
19 Inverse Video
31 Reading Text from Disc or Econet
31 Saving Envelopes
31 Sideways ROM Index Again
35 Hard Spaces in View
35 Noise for Fireworks
38 Better Loading with Tensai Tape Recorders
41 Macros in Assembler
45 Colourful GCOL Commands
45 Reading Sideways ROMs
45 Strange Variables

Tested on Basic I & II and O.S. 1.2 32k

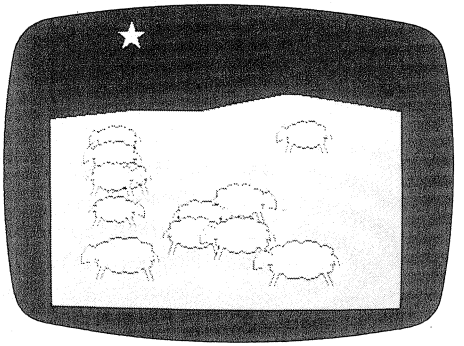
A CHRISTMAS CAROL

by D.G. Chappell

With Christmas approaching we have decided to include a program with a seasonal flavour that provides a good demonstration of the sound and graphics capabilities of your BBC micro. The program builds up an attractive Christmas card display on the screen while playing a rendering of "While Shepherds Watched Their Flocks By Night". We are sure that you will find the results of typing in this relatively short program well worth while. Be careful when copying the program, particularly with the data statements at the end, if you don't want the music to be out of tune.

The program is well structured and you should have no difficulty in identifying the various parts from the procedure names. The music is held as a series of numbers in data statements which are used as the parameters for sound frequency, volume and duration in the procedure PROCMUSIC. Once started the music will continue indefinitely until you press Escape or Break.

The magazine cassette/disc contains an extended version of this program with four complete carols and corresponding screen displays.



```
  10 REM PROGRAM XMAS CAROL
  20 REM VERSION B0.2
  30 REM AUTHOR  D.CHAPPELL
  40 REM BEEBUG  DECEMBER 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 ON ERROR GOTO 350
 110 MODE 1
 120 VDU23,1,0;0;0;0;
 130 DIMQ%(35),R%(35),S%(35),S(7),C(7)
 140 A%=0:I%=20:C(1)=1:C(2)=2:C(3)=3
 150 REPEAT
 160 X%=RND(1279):Y%=RND(1023):Z%=RND(
60)+30
 170 I%=I%-1:IF I%=0THEN I%=RND(25)+5:
REPEAT:C%=RND(7):K%=RND(3):UNTIL(C%<>C(
1)ANDC%<>C(2)ANDC%<>C(3)):VDU19,K%,C%,0
,0,0:C(K%)=C%
 180 PROCSTAR(Q%(A%),R%(A%),S%(A%)+12,
0)
 190 Q%(A%)=X%:R%(A%)=Y%:S%(A%)=Z%
 200 A%=A%+1:IF A%>35 THEN A%=0
 210 UNTIL ADVAL(-6)=15
 220 FOR A%=0TO35:PROCSTAR(Q%(A%),R%(A
%),S%(A%)+12,0):NEXT
 230 RESTORE
 240 VDU26,20
 250 VDU19,1,4,0,0,0:VDU19,2,2,0,0,0
 260 GCOL0,129:CLG:GCOL0,2
 270 MOVE0,650:DRAW300,680:DRAW550,680
:DRAW850,740:DRAW900,740:DRAW1279,690
 280 FOR Y%=0TO740STEP4
 290 PLOT77,900,Y%:NEXT
 300 PROCSTAR(300,950,50,3)
 310 FOR Y%=580TO140STEP-40
 320 PROCSHEEP(150+RND(950),Y%+RND(50)
,2*RND(2)-3):NEXT
 330 C%=1:REPEAT:PROCRAY:PROCMUSIC:UNT
ILADVAL(-6)=15
 340 :
 350 ON ERROR OFF:MODE 7
 360 IF ERR=17 END
 370 REPORT:PRINT" at line ";ERL
 380 END
 390 :
1000 DEF PROCSTAR(X%,Y%,Z%,C%)
1010 VDU29,X%;Y%;:GCOL0,C%
1020 MOVE-0.59*Z%,-0.8*Z%:MOVE0,Z%:PLO
T85,0,-0.37*Z%:PLOT85,0.59*Z%,-0.8*Z%
```

```
 1030  MOVE0.95*Z%,0.3*Z%:MOVE-0.95*Z%,
0.3*Z%:PLOT85,0,-0.37*Z%
 1040 ENDPROC
 1050 :
 1060 DEF PROCMUSIC
 1070 IF ADVAL(-6)<3 THEN ENDPROC
 1080 READ P%,Q%,R%,D%
 1090 IF D%>50THEN PRINTTAB(0,20)P%,Q%,
R%,D%:STOP
 1100 IF D%>1THEN1120ELSE SOUND1,0,0,(1
+30*D%)
 1110 IF D%=1THEN RESTORE:ENDPROC
 1120 SOUND&201,-15,P%,D%:SOUND&202,-13
,Q%,D%:SOUND&203,-12,R%,D%
 1130 ENDPROC
 1135 :
 1140 DEF PROCSHEEP(X%,Y%,H%)
 1150 VDU29,X%;Y%;:E%=40+RND(20)+(13E3)
DIVY%:F%=E%/2+RND(10):L%=11
 1160 X0%=E%:Y0%=-F%/4:MOVEX0%,Y0%
 1170 FOR A=0TO6.3STEP0.15:PROCMUSIC
 1180 X%=E%*COSA+E%/L%*COS(A*L%):Y%=F%*
SINA+F%/L%*SIN(A*L%)
 1190 GCOL0,3:MOVE0,0:PLOT85,X%,Y%
 1200 MOVEX0%,Y0%:GCOL0,0:DRAWX%,Y%
 1210 X0%=X%:Y0%=Y%:NEXT
 1220 PROCMUSIC:FORZ%=0TO16STEP4
 1230 IF Z%=0OR Z%=16THEN GCOL0,0ELSE G
COL0,3
 1240 MOVE0.65*E%+Z%,-0.7*F%:DRAW0.8*E%
+Z%,-1.4*F%
 1250 MOVE-0.65*E%-Z%,-0.7*F%:DRAW-0.8*
E%-Z%,-1.4*F%
 1260 MOVEH%*(-E%+4-Z%),0:DRAWH%*(-E%-Z
%),-F%/3
 1270 NEXT:PROCMUSIC
 1280 GCOL0,3:E%=E%-6:F%=F%/2.3
 1290 MOVEH%*(E%-6),F%:MOVEH%*(E%+F%),8
+F%:PLOT85,H%*(E%-6),0
 1300 PLOT85,H%*(E%+2*F%),0:PLOT85,H%*(
E%+F%),-F%:PLOT85,H%*(E%+2*F%),-6-F%
 1310 GCOL0,0:MOVEH%*E%,F%:DRAWH%*(E%+F
%),8+F%:DRAWH%*(E%+2*F%),0:DRAWH%*(E%+2
*F%),-6-F%:DRAWH%*(E%+F%),-F%
 1320 PLOT69,H%*(E%+F%*1.2),0
 1330 PROCMUSIC:ENDPROC
 1340 :
 1350 DEF PROCRAY
 1360 C%=(C%+2)MOD4:GCOL0,C%:VDU29,300;
900;
 1370 MOVE0,0:DRAW0,-180:MOVE20,10:DRAW
60,-100:MOVE-20,10:DRAW-60,-100
 1380 ENDPROC
 1385 :
 1390 REM ** WHILE SHEPHERDS WATCHED **
 1400 DATA121,101,89,12,137,121,101,18,
137,121,101,6,129,117,101,12,121,109,89
,12,141,109,93,12,141,121,109,12,137,12
1,101,12,129,117,101,12,137,121,101,12,
149,117,101,12,149,129,109,12,145,129,1
09,12,149,117,101,36
 1410 DATA137,121,101,12,157,121,93,18,
149,121,89,6,141,121,93,12,137,121,101,
12,129,117,101,12,121,109,89,12,117,105
,89,12,137,101,89,12,129,117,101,12,121
,109,89,12,121,109,93,12,117,101,81,12,
121,101,89,36,0,0,0,1
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

ACORN TUBE SCREENS - Archie Ewing

If you had a program that used *LOAD or *SAVE to load and save screen memory before you purchased your Tube, then you can still load and save screen memory, but you need to extend the addresses contained. This means using, say FFFF3000 for modes 0, 1 and 2 instead of 3000 (the FFFF says that the operation is to deal with IO processor memory). Z80 users need to extend the command further; viz, **LOAD :0.$.PICNAME FFFF3000. Note that the two ** are necessary, as is the drive specification. The directory is not an essential requisite.

NEW WAY TO CRASH A BEEB? - M.P. Briggs

On disc systems try DIM A%(1,1,1,1,1,1,1,1,1,1,1,1), that's 12 of them, or for tape systems use 14.

RESERVING BASIC II MEMORY

If you are using Basic II and wish to reserve up to 255 bytes of memory when programming in assembler, then you can use the EQUS pseudo command, like this:

```
.memory EQUS STRING$(amount,CHR$0)
```

This also ensures that the memory is cleared to a specific value.

BOOK REVIEWS

With Christmas looming, the ever present problem is here again. We look at a range of suitable stocking fillers in the book world. Presents for relatives, loved ones, or even yourself.

Handbook of Procedures and Functions for the BBC Micro by Audrey and Owen Bishop. Published by Granada at £6.95. Reviewed by Geoff Bains.

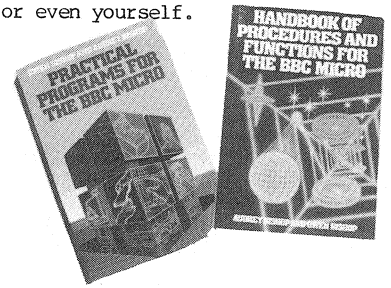
Sounds like a wonderful idea for a book, doesn't it? One book filled the answers to all those little problems that you always come across when writing programs. Unfortunately it's not quite like that.

There are about seventy procedures and functions in this book. Some of them are truly useful - they give you ideas on how to solve trivial problems that are holding up an entire program. None of them are really worth typing in as they stand. There is just too much padding. Most would be more suited as one liners to incorporate in programs, rather than going to all the trouble of a procedure or function.

Some of the programming techniques are downright sloppy - time delays using FOR...NEXT loops. Tut tut.

There are other problems with this book too. The actual program listings are not wonderful. They're rather feint and tatty looking; a great contrast from the clear text and good quality paper. More important is the lack of a useful index. You're not going to read through this book. It is a reference work and the need for an index is clear. However, all that is provided is an alphabetic list of the procedure/function names along with their applications. Surely an alphabetic list of the applications with the choice of program against each would have been more useful.

It's a nice idea, but I'm not really impressed with the result.



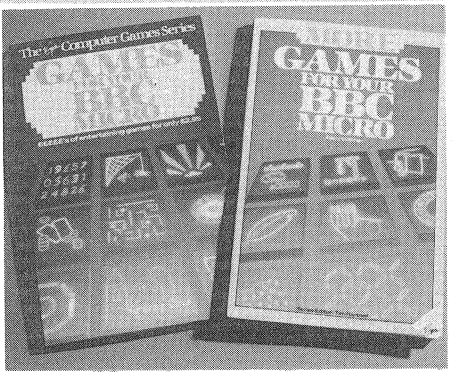
Practical Programs for the BBC Micro by Owen and Audrey Bishop and published by Granada at £6.95.
Reviewed by Mike Williams.

This is yet another new book from Granada, this time concentrating on programs for more serious applications. The book contains listings of 14 complete programs with comprehensive notes on their use and application but little about the programs themselves. Clearly the book is aimed at those trying to find useful tasks for their micro rather than those who want to learn programming techniques. This is an excellent aim, although I do not feel entirely convinced by some of the results.

The chapters, and programs, dealing with cashflow, stock control, accounts and information retrieval are sensible if limited approaches to applications that many may find useful. Many of the other applications such as tipster, pie charts, quiz, phone call coster (!) etc seem to be of much more dubious value. The problem with many of the applications dealt with here is not that the application itself is silly, but that it can only be looked at in a quite trivial way by short programs listed in a book. That is why people are prepared to spend tens if not hundreds of pounds on software for such applications.

If you are particularly interested in one or more of the applications covered by this book, then you will find the relevant sections interesting and useful, particularly if you have the programming skills to develop the printed programs into something really worthwhile. If you are looking for a ready made solution to a problem then you would be better off putting the money towards a good quality software package.

'Games for your BBC Micro' and 'More Games for your BBC Micro' by Alex Gollner. Published by Virgin Books at £2.95 each. Reviewed by Geoff Bains.



These two books each contain about thirty programs, mostly of about 100 lines. As such they are pretty good value. Although the title says that all the programs are games this is not so. Some are amusing graphics displays, something similar to the type that you'll find occasionally in Beebug, and others demonstrations of features of the Beeb.

There is even a chapter, in both books, about how to write your own programs and another giving a limited glossary of computer terms. These are only so much padding, especially as they appear identically in each book. The subject of writing programs is gone into in more detail and better in other books. Let's keep to the title.

All the games are simple. This is an advantage. A book with thirty odd 500 line games may be very clever but is also rather daunting. These programs are short and sharp. The games are either thought games, such as Nim and a reaction tester, or very simple action games ('arcade' is too much an exaggeration). The action games are typically: simple invaders, frogger, and the like.

None of these is going to make the local arcade fearful for its profits, but they do cram a reasonable amount of entertainment into a very few program lines. These books would make great Christmas presents for the very new owner of a BBC micro. It's only a shame, really, that the titles could not have been more accurate.

21 Games for the BBC Micro by Mike James, S.M.Gee and Kay Ewbank published by Granada at £5.95.
Reviewed by Mike Williams.

Although first published some months ago, the prospect of 21 games for your BBC micro at an equivalent price of 30p each (approximately) is obviously still attractive, particularly at this time of year. So let's see how good these games are. The introduction to the book also claims that the programs are intended to improve the reader's knowledge and use of Basic, so the design and structure of the programs must also be considered.

The book itself is generally well produced, the printing including that of the program listings is clear and readable, and the screen shots of the various games have reproduced well (in black and white) though it is sometimes difficult to make much of the display without reading the text as well.

If you expect dynamic, action packed arcade games, sophisticated thought and logic puzzles or engrossing adventure games you will be disappointed. You would probably get better value for money by buying a single commercial game program on cassette. Despite this the games are varied, if somewhat dated now (noughts and crosses, space invaders, horse

racing) and still offer plenty of good, cheap fun for those who are still comparatively new to computing. Most of the programs are of a reasonable length, so not too much effort is needed before you can start playing.

As examples of good programming technique though, these programs certainly leave a lot to be desired, and indeed one might well suspect that many of these games have been converted rather badly from versions written in much more primitive dialects of Basic than that on the BBC micro. Frequent use is made of GOSUB, and not a little of GOTO as well, leading to programs that are difficult to follow, resembling the proverbial plate of spaghetti in the convoluted twists and jumps involved.

I would not recommend this book as any way to learn good programming and I am only moderately enthusiastic about the games, which now seem rather outdated.

The BBC Micro Gamesmaster by Kay Ewbank, Mike James and S.M.Gee, published by Granada at £5.95.
Reviewed by Mike Williams.

I was more impressed by this book than I had expected to be. This is one of the most recent books from the Granada stable for the BBC micro and it shows. The production and presentation



is excellent and the program listings are amongst the best I have seen for clarity of printing. Equally, the authors have exploited all the best features of BBC Basic to present well designed and structured programs of excellent quality.

The purpose of this book is to show those who have already learnt the rudiments of Basic programming how to write good games programs employing animated graphics. Many useful techniques are developed in the course of presenting a total of six complete computer games and the authors have not been frightened to introduce sections of machine code where this is appropriate. Anyone who is keen to develop their own computer games should read this book, but it will have less to offer those who are just looking for a book of listings.

POINTS ARISING

COMPACT FUNCTION KEY DEFINITIONS

The compact definition for the keyword 'AND' should be entered as |!|@ and not simply as described in the original article in BEEBUG Vol.2 No.9. We were also wrong in the update published in BEEBUG Vol.3 No.5 when we said that the additional keywords described required two bytes of memory. Despite the extra characters typed, all the keywords occupy only one byte each in the function key buffer. Our thanks to Mr J.P.Jakubovics for pointing these facts out.

MULTI-SCREEN SLIDE SHOW (BEEBUG Vol.3 No.3)

If you want to use this program to display screens produced using the BEEBUGSOFT Teletext Editor you will need to change the load address for each saved display to &7C00. This can be achieved using *LOAD and *SAVE (loading the screen display to memory and then re-saving), or by using the Disc Snarfer program from BEEBUG Vol.2 No.6. Thanks to the Revd. Melvyn Matthews for providing this useful information.

ested on Basic I & II and O.S. 1·2 16 k

A PAGE OF GAMES TO PLAY

by N. Silver

To keep you amused over the Christmas break, this page contains three complete and challenging games. These games will certainly provide plenty of fun and frustration. Furthermore, each program is written as a single line of Basic, and shows how much can be achieved in such a short space.

The three programs listed here are examples of what can be achieved by using only one line of Basic. Here the programs are extensively abbreviated so that the line will fit into Basic's keyboard buffer. Because of this, you cannot edit a LISTed version, and so, to allow for errors, it is best to spool out a copy of the text to tape/disc initially. This can be achieved as follows:

```
*SPOOL PROGRAM
type in program
*SPOOL
```

Run the program. If there are any errors, *EXEC PROGRAM (rewinding your tape if necessary) and correct them on this version. Once the program is working, it can be SAVEd in the normal manner. With each of the three games, you should press the space bar to restart at the end of a game.

```
1L=0:REP.L=L+3:MO.4:DR.1279,0:DR.1279,4
52:MOVE1279,572:DR.1279,1023:DR.0,1023:
F.I=1TOL:V.31,RND(32)+5,RND(31),42,30:N
.:P.(L-3)/3:X=0:Y=512:REP.PL.69,X,Y:X=X
+4:Y=Y-(INKEY-74+.5)*8:U.PO.X,Y)=1ORX=1
280:U.X<1280:V.7:REP.U.INKEY-99:RUN
```

The first game (called 'Asterisk Tracker') is a very simple game in which you have to guide a 'snake' across the screen, whilst avoiding the stars. As the game progresses, more and more stars will be displayed, and the ease of the game rapidly disappears. The Return key guides the 'snake' upwards, but it moves down if Return is not pressed. Aim your 'snake' for the gap in the far wall, and don't touch any objects as this causes instant death from space acid poisoning!

The second game is very similar to the game 'Truffle Hunt' that we published in BEEBUG Vol. 3 No.2. The object of the game is, with the 'Z' and 'X' keys, to guide your 'snout' up the screen, and to devour as many of the yellow truffles as you can. As you move, avoid the red poisonous mushrooms however, as these prove fatal within microseconds! Your 'snout' will re-appear at the bottom of the screen if it leaves the top, and it comes back on at the other side if you wander too far.

```
1S=0:X=640:Y=0:MO.5:F.I=1TO50:V.5,18;1,
25,4,RND(1270);RND(1023);42,18;2,9,9,12
4,18;3:N.:REP.PL.69,X,Y:DR.X+8,Y:X=X+8*
(INKEY-98-INKEY-67):X=(X+1280)MOD1280:Y
=Y+4:Y=Y MOD1024:P=PO.X,Y):S=S-(P=2):U.
(1ANDP)=1:V.7,4:P.S:REP.U.INKEY-99:RUN
```

The final game is a treasure hunt. Using the standard keys ('Z', 'X' for left and right, '/' and ':' for up and down), guide a small pirate (the '*') around the screen. Two numbers are displayed at the top of the screen. The first indicates the number of moves your pirate has made, and the second the distance that you are away from the object. You should use this number to home in on the treasure, and upon success a short bleep will sound.

```
1@%=778:X=20:Y=11:S=0:D=0:A=RND(40)-1:B
=RND(11):MO.7:REP.C=INKEY9:X=X+(C=90)-(
C=88):X=(X+40)MOD40:Y=Y+(C=58)-(C=47):Y
=(Y+22)MOD23+1:S=S+SGNINS."ZX/:",CHR$C)
:D=SQR(ABS(A-X)^2+ABS(B-Y)^2):V.7807;:P
.S,D:V.31,X,Y,42:U.D=0:V.7:REP.U.GET=32
:RUN
```

Tested on Basic I & II and O.S. 1.2 32k

EXTENDING THE PLOT INSTRUCTION

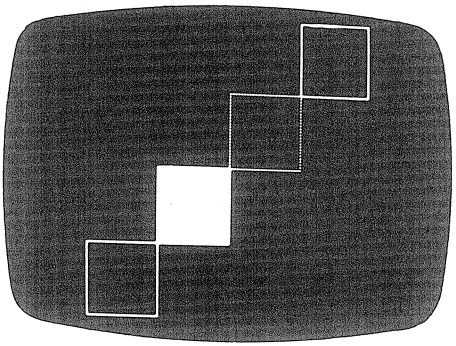
by Alan Dickinson

The description of the PLOT instruction in the User Guide intriguingly describes PLOT codes 32-63 and 88-255 as reserved for future expansions. Alan Dickinson shows how to do this yourself using simple machine code routines to build new PLOT instructions which provide additional graphics functions of your own choice.

Circles? Software sprites? Who knows what Acorn are plotting, (in fact PLOT codes 56-63 have now been used in O.S. 1.2.), but we do have some clues as to how these extensions might be implemented. Using these spare codes enables us to implement additional PLOT instructions which can then be used in any Basic program. Each new function has to be written as a machine code routine, but this is not difficult as the example shows.

There is a little known vector at RAM address &226, and the system performs an indirect jump via this vector whenever an unrecognised VDU command is used; that is when an unrecognised PLOT number is used, or a PLOT command is used in a non-graphics mode, or a VDU23 command is used to reprogram characters in the range 2-31. Remember that PLOT is exactly equivalent to VDU 25 (see page 378 in the User Guide). The 'Carry' flag is clear for PLOT commands, and set for VDU23 commands, so that we can distinguish between the two. For VDU23 commands, the accumulator contains the character number and locations &31C-&323 contain the eight re-definition bytes.

Following an unrecognised PLOT number, the system converts the X and Y coordinates into internal coordinates, taking into account the graphics origin, and whether the PLOT command was using absolute or relative screen coordinates. These internal coordinates are stored in locations &320-&323. The experienced machine-code programmer could make use of this data to produce some very specialised screen output. In the example program, PLOTextend, I have kept things simple, and used OSWORD to convert them back to screen coordinates so it should be obvious what the PLOT commands are doing. As a result this routine is not as fast as it could be.



PLOTextend allows the PLOT numbers 248-255 to PLOT rectangles between the last point visited and the point specified in the PLOT command. Depending on the PLOT number, the rectangle may be a solid outline, a dotted outline, or a filled shape. Like all the resident PLOT commands, it is limited to the graphics window, and obeys the actions specified by GCOL commands. For such a simple graphic it is scarcely worth writing an intercept routine. However, if you were writing a fast and complex routine, perhaps with direct screen addressing, then the PLOT command provides a very neat and powerful interface.

For more information on vectors not documented in the User Guide, see the excellent Advanced User Guide, by Bray, Dickens, and Holmes. Section 10.8 on page 261 describes the VDU (and PLOT) extension vector, known as VDUV, while section 11.4 from page 274 onwards describes the memory locations used to store the various graphics coordinates used in implementing PLOT and other VDU instructions.

The program PLOTextend shows how a new set of PLOT instructions can be

implemented and this serves as a model for any similar extensions that you might program yourself. The program is described in the following notes though you might like to just type the program in (you can omit all the comments) and run it for the sake of its fascinating visual display.

PROGRAM NOTES

The program PLOTextend contains extensive comments and if you work through the program you should have no problem in understanding what each section does. These notes supplement the comments and provide some additional information.

The program consists of the machine code routine that implements the new PLOT instructions (contained in the procedure PROCassemble) and a Basic routine (contained in the procedure PROCdemo) to illustrate the use of the newly defined PLOT instructions.

The start of the program reserves the memory used by the machine code and initialises some important variables. The memory areas are allocated dynamically by the DIM statements in lines 370-400 . They are as follows:

Add.	Len.	Purpose
mc	512	machine code routine itself.
vduv	2	address of the default VDUV extension vector (&FFA6).
pblock	16	current and previous points as screen coordinates.
a	1	temporary storage of accumulator.

The location of these memory areas is not at all critical. The size of the memory area used for the machine code need only be sufficient for the routine being implemented.

The following few lines in the program assign the correct addresses for the two O.S. routines OSWORD and OSWRCH, and the default address for the VDUV vector (&FFA6). This is the address normally stored as the VDUV vector, and the new routines are programmed so that on exit, a jump is made to this default address. These assignments are not essential but the use of names rather than hexadecimal numbers leads to a more readable program.

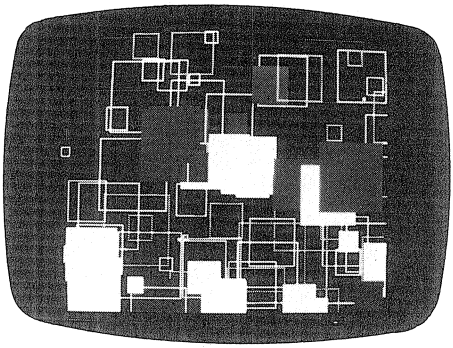
The main program is quite short, from lines 480 to 540. This assembles the machine code, assigns the start address of the machine code to the vector address &226 (in place of the default &FFA6), and then calls the demonstration procedure.

The machine code is well documented and almost self-explanatory. The PLOT number is in the accumulator and it is this which is temporarily stored at address 'a' for later use. A number of checks take place at the start of the machine code before the graphics routine proper is entered. Note how the subroutines 'plot4', 'plot5', 'plot21' and 'plot85' implement the equivalent Basic PLOT instruction, while the subroutines 'x1', 'y1', 'x2' and 'y2' output the appropriate X and Y coordinates as required. At any time an exit is made from the routine it is done by the instruction JMP vduv.

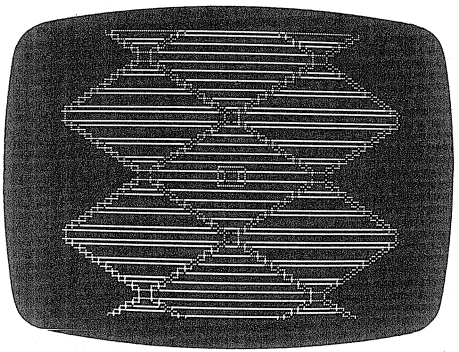
The demonstration program in Basic is visually quite interesting as well as illustrating the use of the new PLOT instructions. It first displays the different types of rectangles to be drawn, then builds up a display of random rectangles, and finally produces a sequence of rapidly changing patterns based again on rectangles. All in all, this is a most impressive display of the new PLOT instructions implemented by this program.

Having seen how a new set of PLOT instructions may be implemented you should now be able to develop other PLOT functions of your own choice.

```
10 REM PROGRAM PLOTE
20 REM VERSION B0.3
30 REM AUTHOR  Alan Dickinson
40 REM BEEBUG  DECEMBER 1984
50 REM PROGRAM SUBJECT TO COPYRIGHT
60 :
100 ON ERROR GOTO 2950
110 REM GRAPHICS EXTENSION
120 REM ------------------
130 REM This routine shows how to
140 REM extend the PLOT commands
150 REM available by inserting a
160 REM machine code routine via ⟶
```



```
170 REM the vector VDUV.
180 REM
190 REM Vector VDUV is invoked for
200 REM VDU23,n commands when n is
210 REM in the range 2-31, or for
220 REM PLOTn commands where n is an
230 REM unimplemented function.
240 REM
250 REM These routines all draw
260 REM rectangles between the
270 REM last point plotted and
280 REM the coordinate specifed
290 REM in the PLOT command, but
300 REM more elaborate code could
310 REM easily be inserted.
320 REM
330 REM PLOT248-251 relative
340 REM PLOT252-255 absolute
350 REM --------------------------
360 :
370 DIM mc 512
380 DIM vduv 2
390 DIM pblock 16
400 DIM a 1
410 :
420 osword=&FFF1
430 oswrch=&FFEE
440 :
450 vduv?0=&A6
460 vduv?1=&FF
470 :
480 FOR opt%=0 TO 3 STEP 3
490 PROCassemble
500 NEXT
510 ?&226=mc MOD 256
520 ?&227=mc DIV 256
530 MODE1:PROCdemo
540 END
550 :
1000 DEFPROCassemble
1010 P%=mc
1020 [OPT opt%
1030 :
1040  \ Carry flag gives
1050  \ reason for entry.
1060  \ Flag SET = VDU23,n
1070  \ Flag CLR = PLOTn
1080 :
1090  BCC lab1
1100  JMP vduv
1110 .lab1
1120 :
1130  \ Accum contains the
1140  \ PLOT number of the
1150  \ unrecognised PLOT
1160  \ command.
1170 :
1180  STA a
1190  AND #&F8
1200  CMP #&F8
1210  BEQ lab2
1220  JMP vduv
1230 :
1240 .lab2
1250 :
1260  \ The routine does not
1270  \ do anything in the
1280  \ non-graphic modes.
1290 :
1300  LDA &361
1310  CMP #0
1320  BNE lab3
1330  JMP vduv
1340 :
1350 .lab3
1360 :
1370  \ &320-3 contain the
1380  \ PLOT coordinates in
1390  \ pixels. These are
1400  \ copied to &314-7, the
1410  \ last-point store.
1420 :
1430  LDA &320:STA &314
1440  LDA &321:STA &315
1450  LDA &322:STA &316
1460  LDA &323:STA &317
1470 :
```



```
1480  \ OSWORD &D returns the
1490  \ current point and last
1500  \ point in screen units
1510  \ which is convenient for
1520  \ use in OSWRCH commands.
1530 :
1540  LDX#pblock MOD 256
1550  LDY#pblock DIV 256
1560  LDA#&D
1570  JSR osword
1580 :
1590  \ The type of draw to
1600  \ be used is obtained
1610  \ from the bottom 2
1620  \ bits of the PLOT num.
1630  \
1640  \ 0 - line
1650  \ 1 - dotted line
1660  \ 2 - solid box
1670  \ 3 - line
1680 :
1690  LDA a
1700  AND #&03
1710  CMP #0:BEQ line
1720  CMP #1:BEQ dotted
1730  CMP #2:BEQ solid
1740  JMP line
1750 :
1760 .line
1770  JSR plot4:JSR x1:JSR y1
1780  JSR plot5:JSR x1:JSR y2
1790  JSR plot5:JSR x2:JSR y2
1800  JSR plot5:JSR x2:JSR y1
1810  JSR plot5:JSR x1:JSR y1
1820  RTS
1830 :
1840 .dotted
1850  JSR plot4:JSR x1:JSR y1
1860  JSR plot21:JSR x1:JSR y2
1870  JSR plot21:JSR x2:JSR y2
1880  JSR plot21:JSR x2:JSR y1
1890  JSR plot21:JSR x1:JSR y1
1900  RTS
1910 :
1920 .solid
1930  JSR plot4:JSR x2:JSR y2
1940  JSR plot4:JSR x2:JSR y1
1950  JSR plot85:JSR x1:JSR y1
1960  JSR plot4:JSR x1:JSR y2
1970  JSR plot85:JSR x2:JSR y2
1980  JSR plot5:JSR x1:JSR y1
1990  RTS
2000 :
2010 .plot4
2020  LDA#25:JSR oswrch
2030  LDA#4 :JSR oswrch
2040  RTS
2050 :
2060 .plot5
2070  LDA#25:JSR oswrch
2080  LDA#5 :JSR oswrch
2090  RTS
2100 :
2110 .plot21
2120  LDA#25:JSR oswrch
2130  LDA#21:JSR oswrch
2140  RTS
2150 :
2160 .plot85
2170  LDA#25:JSR oswrch
2180  LDA#85:JSR oswrch
2190  RTS
2200 :
2210 .x1
2220  LDA pblock+0:JSR oswrch
2230  LDA pblock+1:JSR oswrch
2240  RTS
2250 :
2260 .y1
2270  LDA pblock+2:JSR oswrch
2280  LDA pblock+3:JSR oswrch
2290  RTS
2300 :
2310 .x2
2320  LDA pblock+4:JSR oswrch
2330  LDA pblock+5:JSR oswrch
2340  RTS
2350 :
2360 .y2
2370  LDA pblock+6:JSR oswrch
2380  LDA pblock+7:JSR oswrch
2390  RTS
2400 :
2410 ]
2420 ENDPROC
2430 :
2440 DEFPROCdemo :VDU23,1;0;0;0;0;
2450  :
2460  MOVE32,32
2470  FOR j%=248 TO 251
2480   PLOT j%,200,200
2490  NEXT
2500  :
2510  TIME=0:REPEAT UNTIL TIME>500
2520  :
2530  FOR j%=1 TO 500
2540   GCOL0,RND(3)
2550   X%=RND(1279)
2560   Y%=RND(1023)
2570   S%=RND(32)*8
2580   P%=RND(4)+247
2590   MOVE X%,Y%:PLOT P%,-S%,-S%
2600  NEXT
2610  :
2620  CLS
2630  FOR Y%=0 TO 1023 STEP 128
2640   FOR X%=0 TO 1279 STEP 128
2650    GCOL0,RND(3)
2660    S%=RND(12)+4
2670    FOR k%=0 TO 128 STEP S%
```

```
2680      MOVEX%,Y%:PLOT247+RND(2),k%,k%
2690    NEXT
2700   NEXT
2710  NEXT
2720  GCOL0,3:MOVE0,0:PLOT255,1279,1023
2730  :
2740  TIME=0:REPEAT UNTIL TIME=500
2750  :
2760  REPEAT
2770   CLS
2780   GCOL3,RND(3)
2790   X%=640:x%=X%
2800   Y%=512:y%=Y%
2810   dx%=RND(24)+8
2820   dy%=RND(24)+8
2830   P%=251+RND(2)
2840    FOR j%=1 TO 400
2850     X%=X%+dx%:Y%=Y%+dy%
2860     x%=x%-dx%:y%=y%-dy%
2870     MOVEx%,y%:PLOT P%,X%,Y%
2880     IF X%>1279 OR X%<0 dx%=-dx%
2890     IF Y%>1023 OR Y%<0 dy%=-dy%
2900    NEXT
2910    CLS
2920   UNTIL FALSE
2930 ENDPROC
2940 :
2950   ON ERROR OFF
2960   MODE 7:IF ERR=17 END
2970   REPORT:PRINT" at line ";ERL
2980   END
```

NEWS NEWS NEWS NEWS

YET ANOTHER ROM BOARD

Too late to appear in the review in this issue, Micro-Z has released an external ROM board for the Beeb. Micro-Z claims that this is the first software addressable external ROM board, meaning that the ROM in operation is selected with the normal *FX command and not using a mechanical switch as other external boards do. Connections to the Beeb are made via a small internal board that is plugged into the processor socket in the Beeb which also takes the displaced 6502. The external ROM board also features an expansion socket for further Micro-Z products. These are to include further ROM expansion (available now), RAM expansion and an Eprom eraser/programmer (available early 1985). The ROM board costs £59.95 incl VAT. Micro-Z is on 0392-73662.

PUTTING THE P INTO SECOND PROCESSOR

The popular operating system, UCSDp is now available on the BBC micro equipped with dual disk drives and a 6502 second processor. Developed by TDI, of Bristol, and Acornsoft the P-system includes compilers and utilities for both UCSDp Pascal and Fortran. The complete package will cost you £299. Acornsoft is on 0223-316039.

OF MICE AND MICROS

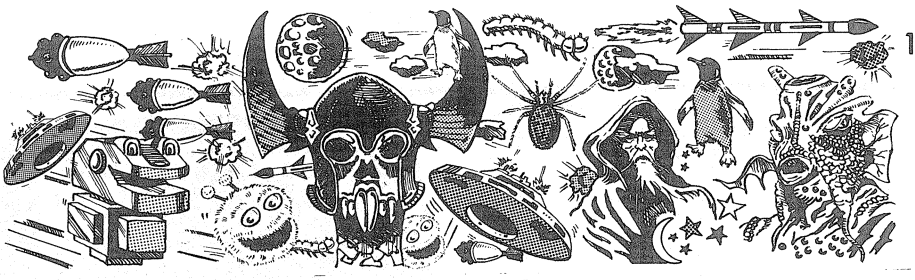
Advanced Memory Systems, one of the first producers of 3.5 in discs for the Beeb has followed another trend with the launch of a mouse and graphics package for the BBC micro. The AmX mouse is a hand held pointing device that is moved around the desk top next to your micro to move a cursor around the screen. The mouse also includes three buttons to select actions. The ArtworX graphics package makes use of the mouse to produce pictures on your BBC micro's screen with the minimum of effort. ArtworX has such features as pull down menus, shading, and so on. It bares more than a passing resemblance to the highly acclaimed graphics package for the Apple Macintosh, a machine costing a little under £2000. AmX and ArtworX are more modest at just £98 incl. VAT for the pair. Further details from AMS on 0925-602690.

QUICK ON THE DRAW

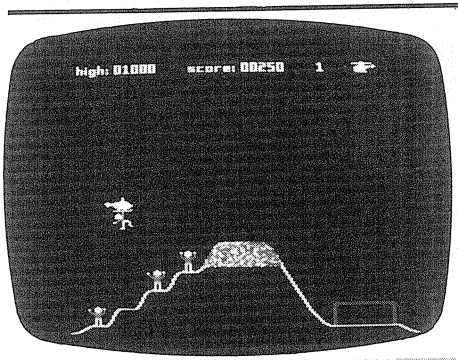
A low cost CAD system is the claim of Ibbotsons Design Software for their DDX package. Costing only £99.95, DDX uses the Beeb in conjunction with disc drives and a Grafpad graphics tablet to produce plans and as a design tool. As well as offering the generation and storage of digitised pictures, DDX features the easy generation of circle, ellipses, and cross hatching. Further details from Ibbotsons on 077-389 658.

ELITIST CLAIMS

Just two weeks after its launch, Acornsoft's Elite game (reviewed in BEEBUG Vol.3 No.6) has sold over 13,000 copies, Acorn claims. By the time that you read this, Acorn expect to have around 100,000 would-be Elitists glued to their screens.



GAMES GALORE



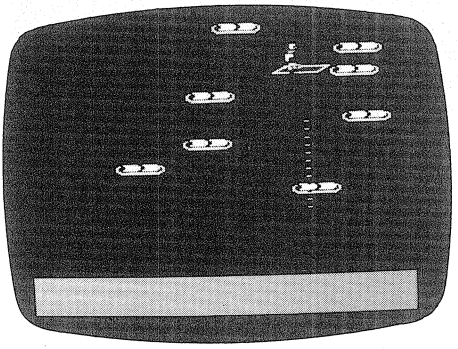
Name : Volcano
Supplier : Acornsoft
Price : £9.95
Reviewer : Geoff Bains
Rating : ***

When Mount Crona (think about it) erupts, there are four people stuck on the hillside just dying to be rescued. Fortunately you happen to have a helicopter handy for that purpose.

The idea is to pick up each of the men in turn, before the lava flows reach him, and take him to the safety of your (apparently fireproof) helicopter base on the other side of the mountain. This would be easy were it not for a few difficulties that nature and Acornsoft put in your way.

Firstly the men can only hang on to the helicopter for a short time. You have to return to base as quickly as possible. This isn't easy as there are boulders exploding out of the volcano. These are not conducive to good flying. It is just as well that your helicopter is equipped with a boulder-blasting gun. More tricky are the ghosts. If one of the four men to be rescued dies (from falling or lava) then he turns into an indestructible, and understandably annoyed, ghost.

Volcano is a very catchy game. The graphics are as quick and smooth as we've come to expect from Acornsoft. Not a blockbuster, but it should run.



Name : Sinbad
Supplier : Virgin Games
Price : £7.95
Reviewer : Geoff Bains
Rating : *

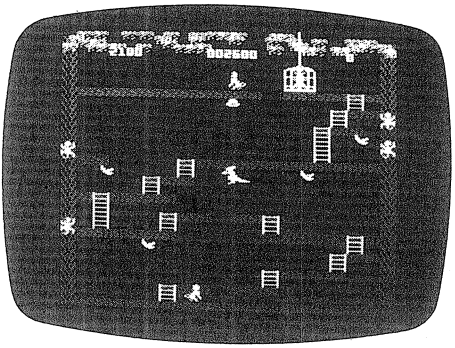
The eighth voyage of Sinbad leaves a lot to be desired when compared to the first celluloid seven. This game is certainly original. That is to say no-one else has bothered to waste time on such an idea.

First the scenario. This is going to sound stupid, but here goes anyway. You are presented with Sinbad's flying carpet swooping around the screen trailing a rope ladder. Also buzzing around are several malevolent life forms. You must jump Sinbad onto the ladder so that he touches these 'life forms' and converts them to peaceful ways. If he touches one already converted he falls off the ladder.

When all that is done it's off for a quick trip through an asteroid storm. If you manage to steer him through that with the awful controls provided, then

you go back to square one with the addition of different life forms and other things like spiders.

Virgin has described this game admirably with its title.



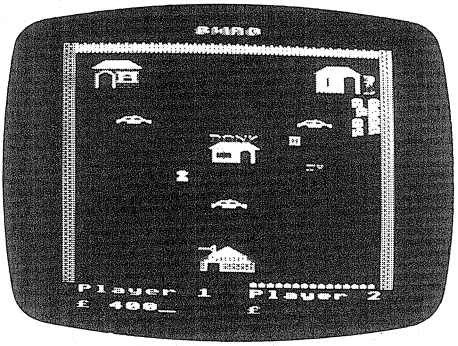
Name : Wallaby
Supplier : Superior Software
Price : £7.95
Reviewer : Geoff Bains
Rating : ****

Another original game, this one, but this time well worth the trouble.

The monkeys have kidnapped (roo-napped?) the baby wallaby and you have to help dad (or mum; it's hard to tell) to rescue it. The baby wallaby is held in a cage at the top of the screen. Up and down the sides and along the various levels, ferocious apple core throwing monkeys roam. These must be avoided as you climb the ladders and hop over the gaps. Fortunately you are equipped with a powerful punch that can dispatch a monkey very nicely to his death.

Just to add to the excitement there are whole apples to gather on the way to give you extra points. No sooner have you rescued the bouncing baby than the whole palaver starts again in a more tricky jungle clearing.

Wallaby is a delight to play. The graphics are excellent, smooth, and cute. The basic idea of climbing ladders and avoiding adversaries isn't new but that doesn't seem to matter. Matilda would have waltzed with joy at this one.



Name : $wag
Supplier : Micro Power
Price : £6.95
Reviewer : Alan R Webster
Rating : ***

$wag is a brand new one or two player game for the BBC micro featuring mode 2 multi-coloured graphics and a practice feature where you can play against the computer.

The game of $wag involves you pitting your wits against the other player to become the first person to steal £250,000 in diamonds. There are several killer droids that try to hamper you in your task of stealing the diamonds. You can also shoot at police cars, but this has the effect of making them angry (the police not the cars), and they then give chase (giving extra money to your opponent). Apparently the only way you can stop these police cars from chasing you is by "...drinking a can of bear..."! (is this a sly advert for a famous lager or just a misprint by Micro Power?)

Overall $wag is a very hectic and enjoyable game with a lot of shooting and action and, at £6.95, gives good value for money.

Name : Super Pool
Supplier : Software Invasion
Price : £7.95
Reviewer : Alan R Webster
Rating : ****

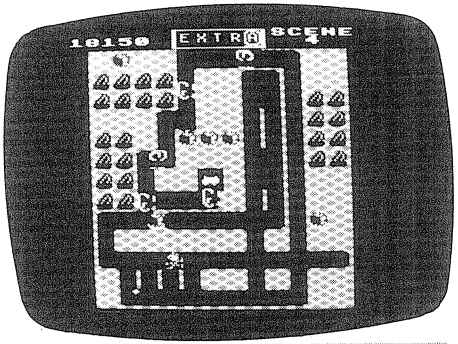
At last, an arcade game with no lasers! Super Pool is a simulation of Pool, the table game, but on a micro.

It's a scaled down version of the real game with just one white cue ball and six coloured balls numbered from 1 to 6.

The object of the game is first to pot all the balls. To do this you are allowed 60 seconds for each shot. Each time you pocket one of the balls, you gain a number of points and these points can be increased if you pot another ball in the same pocket. After potting the first six balls, you proceed to level two, where you have to pot the balls in order, and then level three where the balls have to be both hit and potted in order.

To play the game you have to position a target around the table, moving it clockwise or anticlockwise. Then you choose the strength of the cue ball, signified by a horizontal bar at the top of the screen and when you are ready, release the cue ball which heads towards the target.

The game features excellent graphics, and is both smooth and realistic. This is an excellent example of what can be achieved on a BBC micro with games such as snooker and pool, and is highly recommended if you fancy a break from the 'zapping' noises of other games.



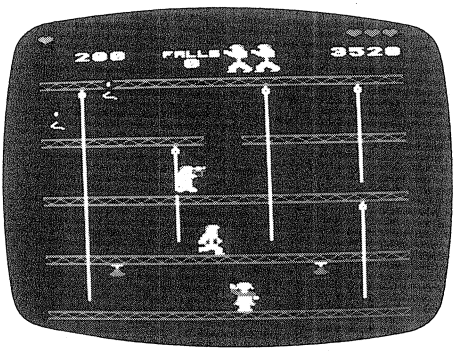
Name : Mr.Ee
Supplier : Micro Power
Price : £6.95
Reviewer : Alan R Webster
Rating : ****

Mr.Ee is an excellent version of the arcade game Mr.Do, and must rate as one of Micro Powers' best releases yet.

In the game you take the part of a wizard whose job it is to collect all of the cherries in the orchard. Unfortunately there are angry 'Umphs' on the rampage which will try to stop you from doing this. To kill the Umphs you can throw your magical crystal ball at them, hoping that it will bounce around the orchard and kill one of them, or you can crush them with apples which fall down when the ground underneath is dug away.

Mr.Ee features ten different screens and has an awful lot happening at once, although, understandably, the game slows down when there are eight Umphs and various other creatures on the screen.

If you are looking for one special game this Christmas, then you should seriously consider this one. Although there is no high score table in the game (due to the amount of memory needed), this shouldn't spoil your enjoyment of such a good game.



Name : Boxer
Supplier : Acornsoft
Price : Tape £9.95, Disc £11.50
Reviewer : David Fell
Rating : **

Boxer is an arcade style game from Acornsoft of an original nature (I'm sure I've heard the various sounds and noises somewhere before, though!). In Boxer, you play the part of an energetic and robust idiot who leaps around a gym attempting to catch five balloons from the poor damsel in distress with the aim of winning her

affection from the other character in the plot, the baddy.

Although described as "fast moving", Boxer does not feature particularly fast graphics, and provided a few hours of frustration before I was able to tolerate the slow response of the game. The graphics themselves are quite attractive, but I don't think these use the available colours as well as they could.

Boxer is an average game that will appeal to some, and create a sense of loathing in others; see it before you buy it!

Tested on Basic I & II and O.S. 1.2 16 k

PRINTER SPOOLER UPDATED

by C.J. Dawson

In BEEBUG Vol.3 No.3 we published a machine code utility that allowed you to print a file whilst carrying on with some other task. The original version did not allow files to be printed from within Wordwise but we have now overcome this with the update described here.

The major part of this update is concerned with allowing the use of the spooler from within Wordwise. This is achieved by taking the original listing, and adding the extra lines below, replacing old lines with the new if necessary. A number of comments are included in the program, but these may be omitted if you wish when you type the program in.

The spooler also now contains a modification to cater for one of the peculiarities of Wordwise. Whenever Wordwise prepares a file with option 8 (spool), it inserts &Ø2 bytes for all of the control codes present within the text file. The updated spooler now checks for these, and sends out a null (&ØØ) byte whenever it encounters an &Ø2. If Wordwise generates one of its own errors (markers or room error), then the spooler will stop operating. This is unavoidable, and is due to the way in which Wordwise works.

There is also now a check for a "£" character, and the user is able to select the character sent to the printer if this is found. Lines 2714 and 2885 should be altered to cater for your own printer. As it stands, it is tailored to a Shinwa printer, but the table below will allow you to configure the program to your own requirements.

One of the limitations of the spooler is that it cannot cope with View, Wordwise and Basic listings if there are any extra codes included such as formatting codes or Basic tokens. The way to over come this is to spool out the text to a file, and to print that file. With Basic this is simply achieved by using the *SPOOL command before listing. This creates a file of pure ASCII codes which can be printed with no problems. Wordwise provides a spool option, and this works quite well (apart from the &Ø2 byte catered for by the update). With View, you need to type *SPOOL and a file name, then SCREEN the appropriate file and *SPOOL again. Having done this you will have a file containing the text correctly formatted, but there will also be the View command mode information. If you wish to delete this, you need to type in NEW, READ filename, delete the unwanted headings, and then SAVE filename. The file can then be printed using the spooler.

EPSON	35
OLIVETTI	35
SHINWA	129

Note: The need to fit the new lines in with the existing program means that the new code is not as readable as we would wish. Take care when typing in these additional lines.

```
  35 REM UPDATES C.J.Dawson.
 160 PRINT"Use *SAVE SPOOLER ";~code;"
";~(P%+7);" ";~(code+&26)
 200 IF code=&900 PRINT"*SPOOLER calls
 the spooler from disc." ELSE PRINT"*RU
N SPOOLER calls from tape"
1085 OSFINDV=&21C:OSBGETV=&216:OSBPUTV
=&218
1225 temposfindv=code+24:temposbgetv=c
ode+26:temposbputv=code+28:spoolflag=co
de+30
1320 P%=code+&26
1351 LDA #0:STA spoolflag
1352 LDA OSBGETV:STA newosbgetvl+1:LDA
 OSBGETV+1:STA newosbgetvl+2
1354 LDA OSBPUTV:STA newosbputvl+1:LDA
 OSBPUTV+1:STA newosbputvl+2
1610 LDA#7:JSR&FFE3:RTS
1811 LDA OSFINDV:STA temposfindv:LDA O
SFINDV+1:STA temposfindv+1 \ Set osfind
 vector
1812 LDA #newosfind MOD 256:STA OSFIND
V:LDA #newosfind DIV 256:STA OSFINDV+1
1813 LDA OSBGETV:STA temposbgetv:LDA O
SBGETV+1:STA temposbgetv+1 \ Set osbget
 vector
1814 LDA# newosbgetv MOD 256:STA OSBGE
TV:LDA# newosbgetv DIV 256:STA OSBGETV+1
1815 LDA OSBPUTV:STA temposbputv:LDA O
SBPUTV+1:STA temposbputv+1 \ Set osbput
 vector
1816 LDA #newosbputv MOD 256:STA OSBPU
TV:LDA #newosbputv DIV 256:STA OSBPUTV+1
2000 BEQ via:JMP exit
2020 .via LDA ifr \ Check printer flag
2040 BNE spooltest:JMP exit \ If not,
done
2045 .spooltest LDA#0:CMP spoolflag:BE
Q disable:JMP printit \ See text
2050 .disable
2290 \ Event 4 always disabled
2300 SEI \ Block interrupts
2302 LDA temposfindv
2304 STA OSFINDV
2306 LDA temposfindv+1
2308 STA OSFINDV+1
2412 LDA temposbgetv:STA OSBGETV:LDA t
emposbgetv+1:STA OSBGETV+1
2414 LDA temposbputv:STA OSBPUTV:LDA t
emposbputv+1:STA OSBPUTV+1
2416 CLI \ Re-enable interrupts
2490 .newosfind \ Test for close files
2495 CMP #0:BNE osfindout \ close a fi
le?
2500 CPY #0:BNE osfindout \ close all
files?
2505 RTS
2510 .osfindout \ Not close all files,
 or any other call
2512 JMP (temposfindv) \ Goto original
 osfind routine
2514 .newosbgetv PHA:LDA#1:STA spoolfl
ag:PLA:.newosbgetvl JSR&FFFF:PHA:LDA#0:
STA spoolflag:PLA:RTS
2516 .newosbputv PHA:LDA#1:STA spoolfl
ag:PLA:.newosbputvl JSR&FFFF:PHA:LDA#0:
STA spoolflag:PLA:RTS
2710 CMP #2:BNE test:LDA #&20:JMP noge
tch \ See text
2712 .test CMP #ASC("£"):BNE nogetch \
 Is it a "£"?
2714 LDA #129 \ Yes see text
2870 CMP #2:BNE test1:LDA #&20:JMP pri
ntit \ See text
2880 .test1 CMP #ASC("£"):BNE printit
\ Is it a "£"?
2885 LDA #129 \ Yes see text
2890 .printit STA ora \ Output byte to
 printer
2965 REM Adjust code location for extr
a code
2970 IF A%=1 OR A%=2 Y%=&C00
2980 IF A%=4 Y%=&900
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

ULA OR SEMI-CUSTOM CHIP?

Acorn have recently started sending out Beebs fitted with a semi-custom chip, as opposed to the video ULA they used originally. This is virtually identical in function to the old ULA. Early batches of this chip required link S26 to be removed, and a wire connecting the centre pin of S26 to IC10 pin 1 to be fitted. If your video ULA needs replacing, and you decide to perform this task yourself, then you will need to fit this if the new chip is one of the early batch. Acorn are now supplying chips that are pin for pin compatible, and which should not have this alteration performed. If you are in doubt, contact your dealer.

INVERSE VIDEO - Ashley Denninson

To produce reverse video without resorting to a sequence of COLOUR statements, use ?&D3=255 to produce inverse video, and ?&D3=0 to revert back to normal video. (Note that modes with more than two colours will need different values - these can be found by experimenting.)

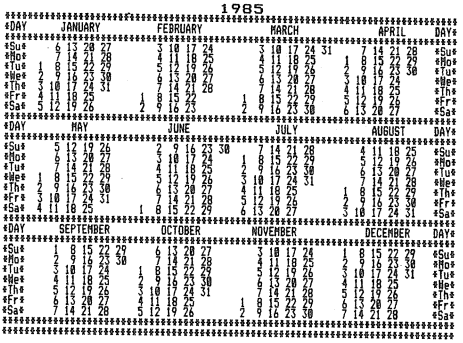
Tested on Basic I & II and O.S. 1·2 32k

CARTOON CALENDAR FOR 1985

by T.A'Hara

What could be better for the forthcoming new year than a calendar with your favourite cartoon character? Using the program listed here you can have just that, and the program can be used to produce calendars for other years too.

This program will in fact produce a calendar for any year after 1980. When you run the program, it asks initially for the year (only integers greater than 1980 will be accepted). In the bottom left hand corner of the screen the word 'DAY' will be displayed with a corresponding number. The number represents the first day of the year with 0=Sunday,1=Monday, etc. From then on all output will be sent to the printer (Centronics compatible). Any special codes included in the listing are designed for an Epson or similar printer, although it can be run without any codes onto any 80 column printer. The codes use REM statements to signify their operation.



1985

DAY	JANUARY	FEBRUARY	MARCH	APRIL	DAY
Su	6 13 20 27	3 10 17 24	3 10 17 24 31	7 14 21 28	Su
Mo	7 14 21 28	4 11 18 25	4 11 18 25	1 8 15 22 29	Mo
Tu	1 8 15 22 29	5 12 19 26	5 12 19 26	2 9 16 23 30	Tue
We	2 9 16 23 30	6 13 20 27	6 13 20 27	3 10 17 24	We
Th	3 10 17 24 31	7 14 21 28	7 14 21 28	4 11 18 25	Th
Fr	4 11 18 25	1 8 15 22	1 8 15 22 29	5 12 19 26	Fr
Sa	5 12 19 26	2 9 16 23	2 9 16 23 30	6 13 20 27	Sa

DAY	MAY	JUNE	JULY	AUGUST	DAY
Su	5 12 19 26	2 9 16 23 30	7 14 21 28	4 11 18 25	Su
Mo	6 13 20 27	3 10 17 24	1 8 15 22 29	5 12 19 26	Mo
Tu	7 14 21 28	4 11 18 25	2 9 16 23 30	6 13 20 27	Tue
We	1 8 15 22 29	5 12 19 26	3 10 17 24 31	7 14 21 28	We
Th	2 9 16 23 30	6 13 20 27	4 11 18 25	1 8 15 22 29	Th
Fr	3 10 17 24 31	7 14 21 28	5 12 19 26	2 9 16 23 30	Fr
Sa	4 11 18 25	1 8 15 22 29	6 13 20 27	3 10 17 24 31	Sa

DAY	SEPTEMBER	OCTOBER	NOVEMBER	DECEMBER	DAY
Su	1 8 15 22 29	6 13 20 27	3 10 17 24	1 8 15 22 29	Su
Mo	2 9 16 23 30	7 14 21 28	4 11 18 25	2 9 16 23 30	Mo
Tu	3 10 17 24	1 8 15 22 29	5 12 19 26	3 10 17 24 31	Tue
We	4 11 18 25	2 9 16 23 30	6 13 20 27	4 11 18 25	We
Th	5 12 19 26	3 10 17 24 31	7 14 21 28	5 12 19 26	Th
Fr	6 13 20 27	4 11 18 25	1 8 15 22 29	6 13 20 27	Fr
Sa	7 14 21 28	5 12 19 26	2 9 16 23 30	7 14 21 28	Sa

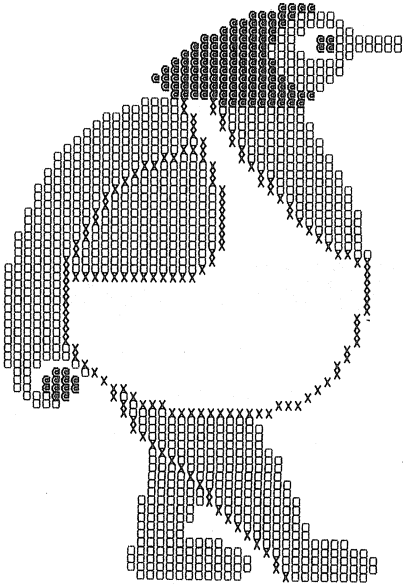
PRINTER CODES

The codes are marked in the listing by appropriate REM statements. For those who may want to change them to suit other printers the codes used are as follows:

15	Condensed
27,51,23	23/216 inch line spacing
27,108,31	Set left margin to 31st column
18	Cancel condensed mode
27,51,18	18/216 inch line spacing
27,108,0	Left margin set to 0
27,71	Set double strike
27,72	Cancel double strike
12	Form feed
27,64	Printer Initalisation

If you know the first day of the year for a particular year (1980 in this program) you can calculate the first day of the year for any year after your starting point by adding days. From this starting point it is then a straightforward task for the program to compile a calendar for the year specified.

Once the calendar is complete the program then draws the well known Trogg



character from the popular game FRAK! produced by Aardvark Software. On non Epson printers an elongated picture will be produced.

The part of the program that draws the cartoon is very simple. It is just a listing of data. The routine itself consists of just two loops, one to READ the data and one to carry out the operation of printing. The data comprises four possible items of information, an integer followed by a character, where the integer is the number of times that particular character is to be printed, -1 to indicate the end of a line, and] (close square bracket) or any other consistent character to indicate a space. Certain characters produce different results in that double strike is used for all characters other than 0. Any number of conditional loops could be added to make use of what ever effects the printer is capable of.

So now you can print out your own cartoon calendar, indeed as many as you wish. Be careful about showing them to your friends though because if our experience is anything to go by they will all want one as well.

If you would like to create your own cartoon character as an alternative to Trogg why not send the result to us and we will award a prize of £50 for the one we judge to be the best. Send a printed copy to the editorial address and mark your envelope 'CARTOON'. Closing date for entries will be 4th January 1985.

The Trogg character from FRAK! is reproduced by kind permission of Aardvark Software.

```
  10 REM PROGRAM CALENDAR
  20 REM VERSION B0.3
  30 REM AUTHOR  T.A'HARA
  40 REM BEEBUG  DECEMBER 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 ON ERROR GOTO 2390
 110 REM SEND CONTROL CODES TO PRINTER
 120 VDU2,1,15,1,27,1,51,1,23
 130 VDU1,27,1,108,1,31,3
 140 DIM Days%(381):DIM Months$(12):DI
M Day$(7):DIM R%(12)
 150 Day=0:CLS:MODE 7
 160 FOR A=1TO2:PRINT CHR$141;CHR$132;
TAB(15)"CALENDAR":NEXT
 170 PRINT'''
 180 FOR A=1TO2:PRINT CHR$141;TAB(5)"E
NTER YEAR OF CALENDAR NOW":NEXT
 190 SOUND 1,-15,20,5
 200 INPUTYear
 210 IF Year<1980 OR Year<>INT(Year) T
HEN SOUND 1,-15,0,5:GOTO 150
 220 PRINT'''
 230 FOR A=1TO2:PRINT CHR$141;SPC(16);
Year:NEXT
 240 PRINT'''''
 250 PRINT"IS THIS CORRECT ? Y/N"
 260 SOUND 1,-15,30,5
 270 Ans$=INKEY$(10)
 280 IF Ans$="Y" OR Ans$="y" THEN320
 290 IF Ans$="N" OR Ans$="n" THEN150
 300 IF Ans$="" THEN270
 310 SOUND1,-15,120,10:SOUND 1,-15,30,
5:GOTO270
 320 SOUND 1,-15,160,3
 330 PROCLEAP
 340 PROCDAYS
 350 PROCCALENDAR
 360 MODE3
 370 PROCPRINT
 380 REM SEND CONTROL CODES TO PRINTER
 390 VDU2
 400 VDU1,18,1,27,1,108,1,0
 410 VDU3
 420 PROCtrogg
 430 END
 440 :
1000 DEF PROCLEAP
1010 T%=Year/4
1020 IF (T%*4)=Year THEN Leap%=1 ELSE
Leap%=0
1030 ENDPROC
1040 :
1050 DEF PROCCALENDAR
1060 FOR T=1 TO12:READ Months$(T),R%(T
):NEXT
1070 D%=Day:FOR T%=1TO12:FOR T2%=1TOR%
(T%)
1080 D%=D%+1:Days%(D%)=T2%
1090 IF Leap%=1 AND D%=59 D%=D%+1:Days
%(D%)=29
1100 NEXT,
1110 FOR T=1 TO7:READ Day$(T):NEXT
1120 ENDPROC
1130 :
1140 DATA JANUARY,31,FEBRUARY,28,MARCH
,31,APRIL,30,MAY,31,JUNE,30
1150 DATA JULY,31,AUGUST,31,SEPTEMBER,
30,OCTOBER,31,NOVEMBER,30,DECEMBER,31
1160 DATA Su*,Mo*,Tu*,We*,Th*,Fr*,Sa*
```

```
1170 :
1180 DEF PROCDAYS
1190 Y%=Year-1980
1200 IF Y%<4 GOTO 1270
1210 Nodays%=Y%/4
1220 Ddays%=Y%-Nodays%*4
1230 Tdays%=Nodays%*1461+Ddays%*365
1240 Fdays%=Tdays%-(Y%*364)
1250 IF Fdays%<7 THEN1270
1260 REPEAT:Fdays%=Fdays%-7:UNTIL Fday
s%<7
1270 IF Y%=0 THEN Fdays%=0
1280 IF Y%=1 THEN Fdays%=2
1290 IF Y%=2 THEN Fdays%=3
1300 IF Y%=3 THEN Fdays%=4
1310 Day=Fdays%+3-Leap%
1320 IF Day>6 THEN Day=Day-7
1330 PROCWAIT
1340 PRINT CHR$133"DAY";CHR$135;"= ";D
ay
1350 ENDPROC
1360 :
1370 DEF PROCPRINT
1380 VDU2
1390 E%=0:EXTRA=0
1400 VDU1,14:PRINTTAB(19)Year:REM DOUB
LE HEIGHT CHARACTER
1410 PRINT STRING$(80,"*")
1420 PRINT"*DAY"SPC(6)"JANUARY"SPC(10)
"FEBRUARY"SPC(12)"MARCH"SPC(14)"APRIL"S
PC(5)"DAY*"
1430 PRINT STRING$(80,"*")
1440 FOR W=1TO7
1450 C%=W+EXTRA
1460 @%=1:PRINT"*";Day$(W);:@%=3
1470 FOR N=1TO4
1480 D1%=0
1490 FOR T=1TO6
1500 D%=Days%(C%)
1510 IF D%=0 THEN PRINTSPC(3);:GOTO 15
70
1520 IF D%>20 AND N=1 AND T<3 THEN PRI
NTSPC(3);:GOTO1570
1530 IF D%<8 AND T=1 AND D%>W THEN PRI
NTSPC(3);:GOTO1580
1540 IF D%>=D1% THEN PRINTD%;ELSE PRIN
TSPC(3);:GOTO1580
1550 IF T<3 AND N=1 AND D%>7 THEN D%=0
1560 D1%=D%
1570 C%=(C%+7)
1580 NEXT T
1590 NEXT N
1600 PRINT"*";Day$(W)
1610 NEXT W
1620 PRINT STRING$(80,"*")
1630 E%=E%+1
1640 IF E%=1 THEN EXTRA=119
1650 IF E%=1 THEN PRINT"*DAY"SPC(8)"MA
Y"SPC(14)"JUNE"SPC(15)"JULY"SPC(13)"AUG
UST"SPC(5)"DAY*"
1660 IF E%=2 THEN EXTRA=238
1670 IF E%=2 THEN PRINT"*DAY"SPC(6)"SE
PTEMBER"SPC(9)"OCTOBER"SPC(9)"NOVEMBER"
SPC(12)"DECEMBER"SPC(4)"DAY*"
1680 PRINT STRING$(80,"*")
1690 IF E%<>3 THEN 1440
1700 VDU3
1710 ENDPROC
1720 :
1730 DEF PROCWAIT
1740 CLS
1750 FOR A=1TO2:PRINT CHR$141;CHR$132;
TAB(15)"CALENDAR":NEXT
1760 PRINT''''
1770 FOR A=1 TO2:PRINT CHR$141;CHR$135
;SPC(4)"WAITING ; OUTPUT TO PRINTER ":N
EXT
1780 PRINT''''''
1790 ENDPROC
1800 :
1810 DEF PROCtrogg
1820 REM **** DRAW TROGG **********
1830 REM SEND CONTROL CODES TO PRINTER
1840 VDU2:VDU1,27,1,51,1,18
1850 VDU1,27,1,108,1,20,1,10,1,10
1860 RESTORE2000:@%=0:REPEAT:READA%,A$
1870 IF A%<0 THEN PRINT" ":GOTO1930
1880 FOR N=1 TOA%
1890 IF A$="]" THEN PRINT" ";:GOTO1920
1900 IF A$<>"0" THEN VDU1,27,1,71 ELSE
VDU1,27,1,72
1910 PRINT A$;
1920 NEXT N
1930 UNTIL A%=-99
1940 VDU1,12:VDU1,27,1,64:VDU3
1950 VDU22,7:PROCAGAIN
1960 IF AGAIN=1 THEN CLEAR:RUN
1970 END
1980 :
1990 REM Top Half 1-6
2000 DATA 28,],4,@,-1,],25,],5,@,4,0,-
1,],23,],6,@,6,0,-1,],22,],6,@,3,0,3,],
2,0,-1,],20,],8,@,2,0,2,],2,@,7,0,-1,],
19,],8,@,3,0,2,],2,@,7,0,-1,]
2010 REM 7-12
2020 DATA 18,],9,@,4,0,3,],2,0,-1,],17
,],11,@,7,0,-1,],16,],12,@,7,0,-1,],15,
],14,@,5,0,-1,],16,],14,@,3,0,-1,],17,]
,15,@,-1,]
2030 REM 13-18
2040 DATA 14,],4,0,1,X,2,],1,X,9,@,-1,
],12,],6,0,1,X,2,],1,X,9,0,-1,],10,],9,
0,1,X,2,],1,X,8,0,-1,],9,],10,0,1,X,2,]
,1,X,9,0,-1,],8,],11,0,1,X,3,],1,X,8,0,
-1,],7,],13,0,1,X,2,],1,X,9,0,-1,]
2050 REM 19-24
2060 DATA 6,],12,0,3,X,3,],1,X,8,0,-1,
],5,],11,0,2,X,2,0,1,X,3,],1,X,9,0,-1,]
,5,],9,0,2,X,5,0,1,X,3,],1,X,8,0,-1,],4
,],9,0,1,X,7,0,1,X,4,],1,X,8,0,-1,],4,]
,8,0,1,X,8,0,1,X,4,],2,X,7,0,-1,],3,],8
,0,1,X,10,0,1,X,4,],1,X,8,0,-1,]
```

EXTERNAL ROM SOCKETS

Reviewed by Geoff Bains

With the plethora of ROMs for the BBC micro, the methods of connecting them abound. In BEEBUG Vol.3 No.6 we looked at the internal ROM expansion boards available. This month Geoff Bains looks at a new trend amongst ROM boards.

If you're blessed with more ROMs than either your Beeb or an internal ROM board can handle, or if you have many that are incompatible, you'll frequently need to swap ROMs in and out of your machine. Of course you can just prise them out of their sockets with the traditional screwdriver or Bic biro top, but this is both cumbersome and, in the long run, damaging. It means taking your Beeb apart every time too.

How much simpler things would be if you could simply plug a ROM into an external socket. Fortunately several companies have also had this thought and there are several external ROM boards now on the market especially designed for the owner who swaps and changes.

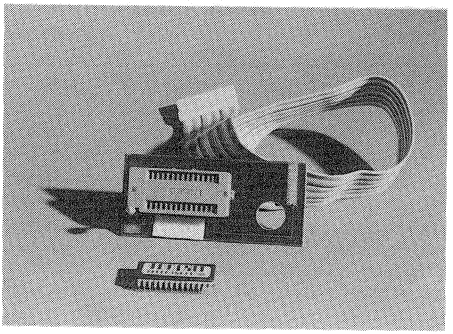
There are problems with all kinds of external ROM socket. The length of ribbon cable that has to come between them and your Beeb circuit board can cause all kinds of strange errors. It is not a definite phenomenon. It happens with some ROMs in some ROM sockets on some Beebs, with some ROM internal boards...in months with an R →

```
2070 REM 25-30
2080 DATA 3,],7,0,1,X,11,0,1,X,5,],1,X
,7,0,-1,],3,],7,0,1,X,11,0,1,X,6,],1,X,
6,0,-1,],2,],7,0,1,X,12,0,1,X,6,],1,X,7
,0,-1,],2,],7,0,1,X,12,0,1,X,7,],1,X,6,
0,-1,],2,],6,0,1,X,13,0,1,X,8,],1,X,5,0
,-1,],1,],7,0,1,X,13,0,1,X,9,],1,X,4,0,
-1,]
2090 REM 31-36
2100 DATA 1,],6,0,1,X,14,0,1,X,10,],1,
X,3,0,-1,],1,],6,0,1,X,13,0,1,X,12,],2,
X,2,0,-1,],1,],5,0,1,X,14,0,1,X,14,],2,
X,-1,],6,0,1,X,13,0,1,X,16,],1,X,-1,],6
,0,14,X,17,],1,X,-1,],6,0,1,X,30,],1,X,
-1,]
2110 REM Bottom Half 37-42
2120 DATA 6,0,1,X,30,],1,X,-1,],6,0,1,
X,30,],1,X,-1,],6,0,1,X,30,],1,X,-1,],6
,0,1,X,29,],1,X,-1,],6,0,1,X,29,],1,X,-
1,],6,0,1,X,29,],1,X,-1,]
2130 REM 43-48
2140 DATA 6,0,1,X,29,],1,X,-1,],7,0,1,
X,27,],1,X,-1,],7,0,1,X,27,],1,X,-1,],4
,0,3,],1,0,1,X,25,],1,X,-1,],1,],2,0,2,
],2,@,1,],1,0,1,X,24,],1,X,-1,],1,],2,0
,1,],4,@,2,],1,X,22,],1,X,-1,]
2150 REM 49-54
2160 DATA 1,],2,0,1,],4,@,3,],2,X,19,]
,1,X,-1,],2,],3,0,2,@,4,],1,X,1,0,1,X,1
7,],1,X,-1,],3,],3,0,6,],1,X,1,0,3,X,11
,],3,X,-1,],12,],1,X,4,0,11,X,-1,],13,]
,1,X,12,0,-1,],13,],1,X,13,0,-1,]
2170 REM 55-60
2180 DATA 14,],1,X,12,0,-1,],15,],1,X,
11,0,-1,],15,],1,0,1,X,11,0,-1,],15,],2
,0,1,X,11,0,-1,],15,],3,0,1,X,10,0,-1,]
,15,],4,0,1,X,9,0,-1,]
2190 REM 61-66
2200 DATA 15,],5,0,1,X,9,0,-1,],15,],6
,0,1,X,8,0,-1,],14,],7,0,1,X,9,0,-1,],1
4,],6,0,2,],1,X,8,0,-1,],14,],6,0,3,],1
,X,7,0,-1,],14,],5,0,5,],1,X,7,0,-1,]
2210 REM 67-72
2220 DATA 14,],5,0,6,],1,X,6,0,-1,],14
,],9,0,3,],1,X,9,0,-1,],13,],12,0,2,],1
,X,10,0,-1,],13,],13,0,2,],1,X,10,0,-1,
],13,],13,0,2,],1,X,10,0,-1,],14,],11,0
,4,],1,X,8,0,-1,]
2230 DATA -99,]
2240 REM END OF DATA
2250 :
2260 DEF PROCAGAIN
2270 CLS
2280 AGAIN=0
2290 FOR A=1TO2:PRINT CHR$141;CHR$132;
TAB(15)"CALENDAR":NEXT
2300 PRINT'''
2310 FOR A=1TO2:PRINT CHR$141;CHR$132;
SPC(11)"AGAIN ?":NEXT
2320 SOUND-1,162,0,0
2330 PRINT'';" (Y/N)"
2340 G$=GET$
2350 IF G$="Y" OR G$="y" THEN AGAIN=1
2360 IF G$=" " THEN2340
2370 ENDPROC
2380 :
2390 ON ERROR OFF:MODE 7
2400 VDU3:IF ERR=17 END
2410 REPORT:PRINT" at line ";ERL:END
```

in their name! Using these extension sockets in conjunction with a well buffered internal ROM board helps to prevent mysterious machine crashes but that is no comfort if this is to be your first ROM board.

Fitting these sockets is not easy either. The bad standard of documentation doesn't help, nor does the fact that they are all designed to connect to one of the four standard ROM sockets in the Beeb and not to an internal ROM board with which they work better!

The external ROM sockets fall broadly into two classes. The first is the simplest and consists merely of a 'zero insertion force' socket that fits in the hole at the side of the keyboard and connects to one ROM position, either on the main Beeb board or on an internal ROM expansion board. The Watford Electronics, and Toad sockets, and ATPL's 'Inside Out' are all of this kind. With all of them the ZIF socket (a special sort of chip socket that will not damage the ROM even on repeated removal and replacement) sits on a small circuit board. From this a ribbon cable connects to a header plug that fits into a spare ROM socket. The cable on the Toad socket is soldered directly onto the ZIF board and directly onto the pins of the header plug. This is not as secure as using a proper connector as the others do.



During the course of this review two of the ribbon cable cores broke at these connections.

The hole at the side of your Beeb's keyboard is not designed to take this kind of socket. It's made for speech vocabulary cartridges (if Acorn ever produces them). This means that the mounting arrangements of the external sockets are not wonderful. Watford Electronics' version is probably the best of its genre. It has a hole on the circuit board through which the mounting pillar for the computer's lid fits. This gives a reasonably stable arrangement. Even then, like all the others it is just stuck to the underside of the lid with double sided sticky tape. The ATPL board has a greater area of sticky tape than most

Product : Sideways ZIF
Supplier : Watford Electronics,
35/37 Cardiff Road,
Watford.
0923-40588
Price : £17.25 incl. VAT

Product : Inside Out
Supplier : ATPL,
Station Rd., Clowne,
Chesterfield, S43 4AB.
0246-811585
Price : £14.95 incl. VAT

Product : ROM Extension Socket
Supplier : Toad Computing,
8 Westbourne Grove, Sale,
Cheshire, M33 1RP.
061-969 4740
Price : £17 approx.incl. VAT

Product : Rom extension unit
Supplier : Ramamp Computers,
25 Avon Drive,
Whetstone, Leicester.
0533-864 966
Price : £48.80 incl. VAT

Product : Rom Box
Supplier : Micro Pulse,
Churchfield Road, Frodsham,
Cheshire, WA6 6RD.
0928-35110
Price : £57.45 incl. VAT

Product : Rom Cartridge System
Supplier : Viglen
Unit 7, Trumpers Way,
Hanwell, W7 2QA.
01-843 9903
Price : £18.95 incl. VAT

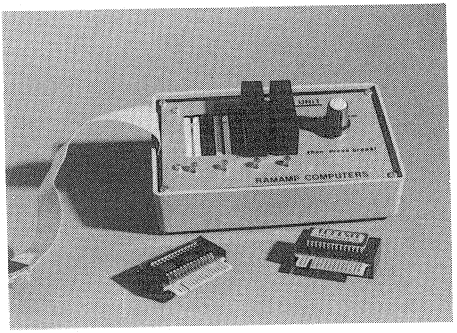


and this makes it firmer in position. In addition ATPL's is not restricted to this position in your machine, but can easily fit on the top, beside it, indeed anywhere.

The other kind of external ROM socket is designed to take several ROMs just like its internal counterpart. Here, however, the selection of which ROM is in operation does not depend on software. A switch on the unit takes care of this.

The idea behind this blatant disregard for Acorn's clever operating system is so that you can change over ROMs without knowing the fancy *FX commands.

Ramamp's and Micro Pulse's ROM boxes are of this type. The Micro Pulse version has room for eight ROMs that plug into normal chip sockets under a transparent lid. A switch on the side selects the ROM in operation. Ramamp's unit has four sockets for special cartridges. These are in effect small circuit boards containing a chip socket



inside a plastic case. These plug into the ROM box.

The advantage of this system is that not only can you select the ROM of your choice from those in the box, but the range in the box can be easily changed. The drawback is that the spare cartridges, essential to make full use of the idea, cost a rather staggering £3.80 each.

Different from either of these types of external ROM socket, but combining many of the advantages of both, is the ROM cartridge system from Viglen. This gives you a special socket that fits into the hole at the side of the keyboard. Into this fit cartridges containing ROMs.



The socket fits positively into the keyboard hole. It has been specially moulded for this position and so can be both inserted securely and removed easily. It is not a botched up, off the shelf answer. The cartridges, too, are made well. The problem with this system is, again, that of price. Producing special plastic mouldings of this kind is not cheap so the system costs £19 for the initial pack and then nearly £7 for each empty cartridge. This is a great shame as this effectively rules out an otherwise excellent idea.

Of the rest, if you want the switchable option and don't mind paying over the odds for the cartridges then the Ramamp ROM box is probably your best bet. Otherwise, a single ZIF socket type would seem the answer and ATPL's and Watford's the better of these.

Tested on Basic I & II and O.S. 1-2 16k

IMPROVED TRACE FACILITY

by Martin Dale

This month we present a useful utility that extends the Basic trace facility by printing the line number in the top left hand corner of the screen, avoiding a screenful of scrolling numbers, and by providing a delay between each line executed.

Most serious program writers find the occasional need for a trace facility when debugging programs. Unfortunately, the original facility in BBC Basic will, unless used very selectively, fill the screen with an horrendous and confusing amount of output. The short utility listed here is designed to extend the Basic trace to overcome these drawbacks.

There are two basic improvements implemented in this extension; the printing of the line numbers only in the top left hand corner of the screen (or current text window), and the facility to pause between each line to allow you time to study the current position in the program.

To use the program, type it in and save it to tape or disc (it is sensible to save it first in case there are any errors in the machine code). Once run, the extended trace facility will initialise itself, and you may then use it as you wish. Note that it will be lost on pressing Break, and that it shouldn't be initialised twice or the machine will hang. As with the standard Basic trace, you use TRACE ON to enable it, and TRACE OFF to disable it. See page 367 of the User Guide for more details on the TRACE command.

HOW IT WORKS

The utility works by intercepting the operating system routine OSWRCH, which writes a character, and performing three tests to check for the start of printing of the current line number. These tests are: is the trace flag set, is this character not within a VDU sequence, and is it the "[" character (the trace facility prints line numbers within square brackets)? If all of these tests are true, then the extended trace routines are called.

The current cursor position is first stored away, as are flags indicating whether the text and graphics cursors are joined and that the trace routine is active. Next, the cursor is positioned at the top left of the screen (more accurately, the current text window). Once this is done, the routine exits back to Basic, which then prints the current line number.

When the printing of the current line number is complete, the trace routine is again entered (this is triggered by the trailing space after the line number). A simple loop will normally wait for 2 seconds, but can be made to terminate quicker by the press of a key, or immediately by a press of the Shift key. This is designed to allow the program to be stepped through at a variety of speeds, allowing correct sections to be passed over very quickly whilst up to 2 seconds can be spent on any lines which seem worthy of more detailed study.

Once the delay has finished, the cursor is restored to its original position, the trace active flag is cleared, and the text and graphic cursor are re-joined if necessary. The routine then exits, and the user's program carries on running as normal.

KNOWN LIMITATION

There is a known limitation in the current version of the program; any attempt to print a "[" as a single character (ie not as part of a graphics sequence) whilst tracing will cause the program to respond un-predictably. There is no easy way to cure this, but knowing that it exists should help users to avoid too many problems. BEEBUGSOFT's Sleuth, a powerful Basic monitor, includes a similar trace option, amongst its many other features, but without this limitation.

→

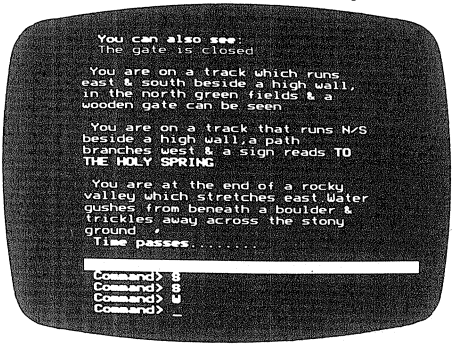
```
10 REM PROGRAM TRACER
20 REM VERSION B0.01
30 REM AUTHOR MARTIN DALE
40 REM BEEBUG DECEMBER 1984
50 REM PROGRAM SUBJECT TO COPYRIGHT
60 :
100 osbyte=&FFF4
110 FOR pass=0 TO 2 STEP 2
120 P%=FNlocate
130 [OPT pass
140 .newvec
150 PHA
160 LDA &20          \ Get trace flag
170 BNE trcon        \ Branch if <> 0
180 PLA
190 JMP printchar
200 .exit
210 PLA
220 .exit2
230 LDX temp         \ Restore
240 LDY temp+1       \ registers
250 .printchar
260 JMP (oldvec)     \ Print character
270 .trcon
280 STX temp         \ Save registers
290 STY temp+1
300 LDA #&DA         \ Is it in the
310 LDX #0           \ middle of a
320 LDY #&FF         \ VDU command?
330 JSR osbyte
340 TXA
350 BNE exit         \ Return if yes
360 LDA trcflg       \ In trace output?
370 BNE midtrc       \ Yes, branch
380 PLA
390 CMP #ASC("[")    \ Start of trace?
400 BNE exit2        \ No, return
410 INC trcflg       \ Set trace flag
420 LDA #&86         \ Save cursor
430 JSR osbyte       \ position
440 STY vpos
450 STX pos
460 LDA #&75         \ Is VDU 5 set?
470 JSR osbyte
480 TXA
490 AND #32
500 BEQ ntgraphic    \ No, branch
510 INC trcflg+1     \ Yes, set flag
520 LDA #4           \ VDU 4
530 JSR printchar
540 .ntgraphic
550 LDA #30          \ Move cursor to
560 JSR printchar    \ top left corner
570 LDA #ASC("[")    \ Print left "["
580 BNE printchar    \ and return
590 .midtrc
600 PLA
610 CMP #ASC(" ")    \ Finished ?
620 BNE exit2        \ No, return
630 JSR printchar    \ Clear "]"
640 DEC trcflg       \ Restore flag
650 LDA #31          \ Restore cursor
660 JSR printchar
670 LDA pos
680 JSR printchar
690 LDA vpos
700 JSR printchar
710 LDA trcflg+1     \ Cursors joined?
720 BEQ ntgrphic     \ No, branch
730 LDA #5           \ Rejoin cursors
740 JSR printchar
750 DEC trcflg+1     \ Clear VDU 5 flag
760 .ntgrphic
770 LDA #21          \ Flush keyboard
780 LDX #0           \ buffer
790 JSR osbyte
800 LDA #200         \ 200 ticks
810 STA delay        \
820 .loopdelay       \
830 LDA #129         \ Shift key
840 LDX #255         \ pressed?
850 LDY #255         \
860 JSR osbyte       \
870 CPY #255         \
880 BEQ exitdelay    \ Yes, exit
890 LDA #129         \ No, wait a
900 LDX #1           \ tick. Key
910 LDY #0           \ pressed whilst
920 JSR osbyte       \ waiting?
930 BCC exitdelay    \ Yes, exit
940 DEC delay        \ No, any more
950 BNE loopdelay    \ ticks?
960 .exitdelay       \ exit delay
970 RTS
980 .delay           \ delay byte
990 NOP
1000 ]
1010 trcflg=P%
1020 temp=P%+2
1030 pos=P%+4:vpos=P%+5
1040 oldvec=P%+6
1050 NEXT
1060 !P%=0:!(P%+4)=0
1070 ?oldvec=?&20E:?(oldvec+1)=?&20F
1080 ?&20E=newvec MOD 256
1090 ?&20F=newvec DIV 256
1100 END
1110 :
1120 DEF FNlocate
1130 LOCAL A%,X%,Y%
1140 A%=0:Y%=0
1150 A%=(USR&FFDA)AND&FF0000 DIV&10000
1160 IF A%=1 OR A%=2 X%=&D00
1170 IF A%=4 X%=&A00
1180 IF X%=0 PRINT"Tape or Disc only!"
:END
1190 =X%
```

ADVENTURE GAMES

by Mitch

A midnight raid on the dwarfs' repository unearths some new adventure games for the Beeb. Wizard Mitch tests the pen's might against the sword's.

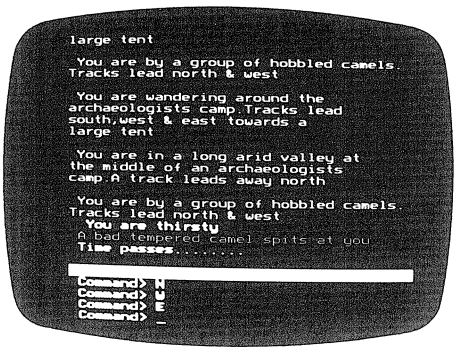
Pausing only to grab a couple of rejuvenation spells, I left the dragon to guard the dungeon and conjured myself inside the first offering.



Title	:	Sadim Castle
Supplier	:	MP Software Ltd. 165 Spital Rd. Bromborough, Merseyside
Price	:	£7.50 cassette. £10.50 disc.

Having been foully murdered by her husband, the Lady Leonara has cursed the castle and all who set foot within it. It was my task to find her mortal remains and lay them to rest in peace. The present ghastly inhabitants didn't seem impressed by my chivalry, however, and did their damnedest to kill me. A few kindly souls did assist me, if I first solved their problems, so it does help to be polite! The game is a coloured text, split screen affair with lots of well written descriptions. The castle abounds with locked doors which require an endless supply of keys. Opening doors becomes such a confusing business that I found I spent an hour trying to unlock one which I later found wasn't even locked!

Not so easy is the solution to the wandering monster that follows you. Unfortunately he continues to reappear with monotonous regularity causing you to repeat the same action ad-infinitum. The game however has a nice 'feel' to it and is not too difficult.



Title	:	The Valley Of The Kings
Supplier	:	MP Software Ltd. 165 Spital Rd. Bromborough, Merseyside
Price	:	£7.50 cassette. £10.50 disc.

A similar format of coloured text and split screen, this game found me crawling around in decaying corpses beneath the pyramid of old King Tut. The place is so hot and smelly that you'd better ensure you bring lots to drink and something to keep the deadly stink out of your nostrils. I found a laundry basket which hissed and a pit in which something nasty was slithering!

A real time element coupled to your endless thirst and flickering torch has

been added to the game. You have no time to stand and stare but must hurry ever forward towards that light at the end of the tunnel. Which knowing my luck will be a train coming the other way!

This game has all the usual ingredients, but in that is its weakness. I couldn't find that magic something which is needed to keep me battling forward so I folded my tent, mounted my camel and stole off home.

Among this month's offerings I have a book as well:

> How To Write Adventure Games by Peter Killworth published by Penguin Acorn Computer Library at £5.95.

I confiscated this from the Troll. He is trying to write an adventure in which he gets to keep the gold for a change.

The book contains many of the secrets used in the Acornsoft games, in fact most of the procedures are a straight copy from Philosophers Quest. Topics include how to write 'Hack and Slash' games, a full-size Roman-style adventure, a chapter on Databases and all the procedures needed to create your own games. I give this book four stars for the clear and amusing style, but I deduct one as some Demon Editor failed to notice that part of the program listing on page 133 has not been printed. Luckily the corrupted lines are used elsewhere in the book (page 127), so all is not lost. However I hurled a few of my more dreadful curses before finding those.

Having earned my supper by solving that problem I highly recommend the book as essential reading by all fans. The infectious enthusiasm of the author is so strong it is bound to inspire any troll to write his own game. Trouble is, if all wizards give away secrets like Peter, this dungeon is going to get pretty crowded!

FLOPPY TAPE DRIVES

Reviewed by Geoff Bains

Many BBC micro users, despairing of their lethargic cassette recorders, will still think twice before spending the large amounts needed for a disc system. Geoff Bains investigates an alternative.

Product : Ultradrive
Price : £83.40
Supplier : Ikon Computer Products
Kiln Lane, Laugharne,
Dyfed, SA33 4QE.
Ø99421-515

Product : Phloopy
Price : £147.75
Supplier : Phi Mag Systems
PO Box 21, Falmouth,
Cornwall, TR11 3TD.
Ø326-76Ø4Ø

With a disc drive interface for your BBC costing upwards of £7Ø and the drives themselves priced at over £1ØØ, the atmosphere is ripe for an alternative system. Ikon Computer Products and Phi Mag Systems both have cheaper alternatives that claim to perform like disc drives.



ULTRADRIVE

The Ultradrive is an improved version of the Hobbit floppy tape system reviewed in Beebug Vol.2 No.2. The Ultradrive records programs and data onto a standard Philips mini

cassette, as used in many dictation machines. The mini cassette deck is totally under the control of the Beeb.

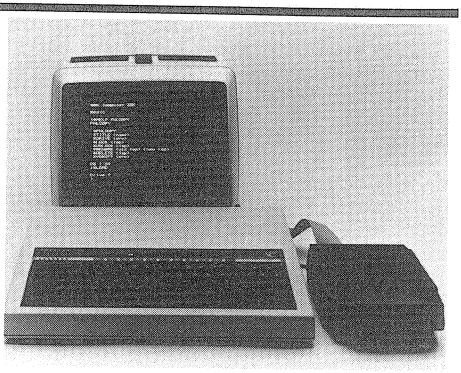
Fitting the Ultradrive to your BBC is simplicity itself. The actual drive comes in a (cream!) case about four inches cubed. This connects to the user port and the auxiliary power socket under the computer by means of a couple of cables. Once that is done there is only the inevitable control ROM to fit.

The Ultradrive works in two modes. In the simple mode the Ultradrive acts as a very fast cassette recorder with no need for manual control nor any of the Beeb's memory. In the complex mode the Ultradrive takes on more of the facilities of a disc drive. Now up to five files can be open at once and two drives can be used though the penalty for this is that PAGE is set to &1600.

Depending where you buy them, the cassettes cost up to £3.50. Each can store about 100K of data. This is arranged in four sections. There are two 'sides' to each tape and each side has two separate directories. Because of the Ultradrive's cassette heritage, all files must have a length of tape (in blocks) assigned to them before they are first stored. This can be inconvenient when you have really no idea of the final length of a program that you are working on.

There is a very laudable system to warn you that you're about to overwrite one file with another of the same name, and give you the chance to change your mind. In addition files can be locked with a *ACCESS command just as they can with the DFS, though the utility to do this is unfortunately on a separate utilities cassette. There are plans to bring out a ROM upgrade, containing all the utilities, for about £20.

The Ultradrive filing system is a clever piece of software that makes the most of the limitations of this cassette system. Most of the disc filing system commands are supported and, for the machine code programmer, most of the disc operating system calls are also there for the using.



Although the facilities for random access are there, the time taken for the Ultradrive to rewind to the required record can mean that sometimes this is not practical. The Ultradrive will probably be used by most Beeb owners only as a super-fast cassette recorder.

However, as such, the Ultradrive scores well. A commercial game that takes 3 minutes to load from normal cassette takes a mere 40 seconds from the Ultradrive.

Unfortunately there is next to no software available in the mini cassette format. Any game that you want to load from the Ultradrive (in double quick time) you first have to transfer from cassette. Not an easy task for many. However, a supplied utility will transfer many cassette programs onto an Ultradrive tape. If you get really stuck then Ikon promises to transfer any software sent in, for free.

PHLOOPY

Phi Mag Systems have scorned the 'off the shelf answer' to produce a custom designed floppy tape drive. The Phloopy is much prettier than the Ultradrive but nearly twice the price. The Phloopy uses continuous loops of tape housed in a plastic cartridge about the size of a cigarette case. The cartridges will store 100K and cost £20 for a pack of five. A demo cartridge and a blank one are included in the starter pack. Inside the smart black drive there is a nine channel tape head and a complete microprocessor based control system. You get a lot of sophistication for your money.

The Phloopy is also a much more complicated add-on to fit than the Ultradrive. There are two small header plugs and a small circuit board that fit into the Beeb's disc interface sockets. The Phloopy drive itself then plugs into the disc drive socket under the Beeb and takes its power from the auxiliary power socket. You can't have a Phloopy and a disc drive interface in your machine at the same time, but you can have up to eight Phloopies fitted.

There is also a control ROM to be fitted, of course, but fitting doesn't end there. You also have to cut two resistors on the Beeb's own circuit board. This makes restoring your Beeb to normal, should you want to upgrade to discs later, very much more difficult.

Once installed, you can treat your Phloopy much like a disc drive. The 'Loop Filing System' that accompanies the drive is, to the user, as near to the DFS as can be. The *ACCESS command is for some reason called *LOCK and *UNLOCK for its two actions. This is not only an annoying difference, but a confusing one, considering Acorn's own 'locking' of files with a quite different meaning.

Random access filing is catered for very nicely on the Phloopy, with all the commands that you'd expect to find on a disc system. However programs that make extensive use of random access are tediously slow. This is because the tape loop inside the Phloopy cartridge is continuously moving round. Once one item of data has been read in, if you want to read another from the same file, you have to wait until the same area of tape comes around again.

The loop system does have the advantage that no slow rewinding is needed along the length of the tape, as the Ultradrive requires. Loading a single program can therefore be much quicker. It's all swings and roundabouts, but the overall advantage seems to be with the Ultradrive.

One advantage, for now, that the Phloopy has is that all the utilities it uses are in the ROM. Like the Ultradrive, these include a program to transfer software from cassette. Phi Mag Systems is (so far unsuccessfully) trying to persuade software houses to use Phloopies.

Although the Phloopy is, in many ways, a more elegant solution to the problem, it lacks the reasonable price and the well thought out approach of the Ultradrive. If you want a flashy add on to impress your friends, the Phloopy wins hands down. If however you are looking for a useful stopgap between cassette and discs then the Ultradrive is not only the cheaper solution but more practical one as well.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

READING TEXT FROM DISC OR ECONET - D.J. Scott

If you are writing a program that is going to be displaying a large amount of text, and you are using disc or Econet, then a considerable amount of memory can be saved by using *TYPE in the program to print out files of information.

SAVING ENVELOPES - J.S. Swiszczowski

In the past we have published hints on how to save definitions for function keys and user defined characters. This basic technique can be extended to save the currently defined envelopes. This is accompilished with:

```
*SAVE ENVS 8C0+100
```

SIDEWAYS ROM INDEX AGAIN - P.J. Higgins

If you have been using the ROM lister published in BEEBUG Vol.2 No.9 with Printmaster fitted, then you will find that because of its length the title will be concatenated with the next one. The solution to this is to add an additional line 1525 LDA &74 and alter the #19 to #20 in line 1530.

BEEBUG Workshop

INDIRECTION OPERATORS (PART 1)

by Surac

This month we start a two part series on the use of indirection operators with a general introduction to the subject, and a detailed look at the use of "?" and "!" in manipulating numbers in memory.

Indirection operators allow a Basic program to access directly locations in memory. The necessity for this occurs when using some operating system calls, and they are very useful for creating some forms of data structure and manipulation. Those of you who are familiar with other dialects of Basic will come to realise that indirection operators are roughly equivalent to PEEK and POKE in their function.

To illustrate the use of indirection operators, and expose their limitations, we'll use one byte of memory, say &70. An understanding of hex helps when using indirection operators. Note that memory locations &70 to &8F are always free to a user program. We'll then perform a number of fairly simple actions upon this location, and observe the results. Type in the following, and study the results displayed.

```
A=10 : PRINT A
?&70=10 : PRINT ?&70
```

The "?" indirection operator is normally pronounced as 'query', although you may find it helpful to read it as meaning 'the contents of'. The name itself allows us to distinguish between the different indirection operators. In both cases above, the same result is printed; ie 10. Now change the 10 to a 1000, and try the examples again. In this instance, different results will be produced. What is happening is that the "?" operator only works with one byte of memory at a time. The number 1000 is bigger than 255 (the maximum number which can be stored in a byte) and thus the low byte only of this number (232) is assigned to address &70 (see diagram below). Clearly, if we want to store numbers bigger than 255, the "?" operator is not appropriate.

Decimal number	1000
Binary equivalent	00000011 11101000
Hex (low byte)	&E8
Decimal (low byte)	232

Fortunately, there is a method of storing numbers up to +/- 2,147,483,647. This is accomplished by using the "!" operator (pronounced pling), which operates on four bytes at a time. Thus, if you change the "?" characters to a "!" characters in the above examples, and then retype them, you will find that the 1000 is correctly handled. Another limitation of the "?" operator is that it does not handle negative numbers as fully as "!" and normal Basic variables. The "?" operator can only handle negative numbers which have already been converted to a 2's complement format (see the Advanced User's Guide, page 31).

To illustrate another point, change the number being using to 34.34 instead of 1000, and try the above examples again. This shows that pling stores numbers only as integers (ie no decimal part). There is no indirection operator that copes correctly with floating point values.

In summary, "?" addresses a byte at a time, whereas "!" works with four bytes. Both deal just with integers, but only "!" copes with negative numbers in the way that you are used to, and thus it is the use of "!" which is equivalent to the use of a normal integer variable in Basic. →

One immediate use of the "?" operator is to display in detail the contents of any section of memory. The program below demonstrates the use of "?" in this way.

```
10 FOR A%= PAGE TO TOP
20 B%=?A%
30 PRINT ~A%,~B%,"=>";
40 IF B%<32 OR B%>126 B%=42
50 VDU B%
60 PRINT
70 NEXT
```

This steps through the area of memory holding your program and reads the byte at each location into B% (carried out by line 20). It then prints the memory address and the byte value in hex, and checks to see if the byte is printable as an ASCII value. It sets B% to the ASCII value of an asterisk if it isn't because various control codes could trigger off the drawing of random graphics in the middle of the program. The byte is then printed, and the loop is continued. If you run this program straight away, the characters displayed will be those corresponding to this program. You could also program a function key to do this and use the routine to look at any Basic program you choose to load into memory. Note the way in which the variable A% holds the address of the byte of memory which we wish to access.

The program below illustrates a slightly more powerful way of using the indirection operators. See if you can anticipate the function of line 90 before reading any further. When you are ready, type in the program, run it, and then press a few keys on the keyboard (you should make sure that you are not in mode 7 before you run the this program).

```
10 DIM X% 20
20 Y%=X% DIV 256
30 A%=10
40 REPEAT
50 ?X%=GET
60 CALL &FFF1
70 VDU 23,128
80 FOR I%=8 TO 1 STEP -1
90 VDU I%?X%
100 NEXT
110 VDU 128
120 UNTIL 0
```

This short program uses one of the operating system calls known as OSWORD (at address &FFF1) to read the definition of a character, and then manipulates this data to write the character upside down. Note the way in which line 10 is used to allocate 21 (0 to 20) bytes to the program, with the start address (of the first byte) assigned to X%. The CALL to OSWORD (see the User Guide page 214 for more information on the use of CALL) requires that the 'high byte' of the address be in Y%, and this is achieved by line 20 in the program. Although X% contains the full address the OSWORD call takes only the low byte of this value. This manipulation of an address in X% and Y% occurs frequently when calling operating system routines.

Line 90 of the program contains an an interesting variant on the use of "?": I%?X%. The effect of this is to access the contents of (X%+I%). This provides a powerful and flexible use of "?" that has some similarities with Basic arrays. By setting X% to point to the start of a set of bytes and incrementing I% within a loop, it is easy to reference each byte in turn, as in line 90 above. The alternatives to this would be to use ?(X%+I%), or for the loop to go from X%+8 to X%+1, both of which are less elegant.

After calling OSWORD at line 60 with A%=10, the eight bytes defining the characters will be stored from the address specified by X% and Y%. By outputting these bytes in reverse order (lines 80 to 100) the character is displayed upside down on the screen. A further example of the extensive use of indirection operators occurs in the movedown routine published in BEEBUG Vol.3 No.5.

Next month we will conclude our look at indirection operators by looking in particular at their use in connection with string handling.

SOUNDING OUT THE BEEB

Two Books Reviewed by Steve Ibbs

Two books have recently been published dealing with music and the BBC micro. We asked Steve Ibbs, a lecturer in music and the director of an electronic music studio to read both and give his reaction to them.



Making Music on the BBC Computer by Ian Waugh published by Sunshine Books at £5.95.

This book contains 244 pages divided into 13 chapters, and all the programs are conveniently available on cassette if required for a further £5.95.

The first chapter of this book is an easy guide to 'What is Sound', and how to generate sound on the BBC micro. It is sufficiently detailed, without losing those very readers the book was aimed at, namely the enthusiast with perhaps not much theoretical expertise. The next chapter then proceeds to explain traditional musical notation with staves, scales, and rests.

Chapters four and five examine the SOUND and ENVELOPE commands in some detail. There are nine sample envelopes for various instruments and the accompanying program displays the envelope, and allows the user to change it very easily and hear the result. Another program later in the chapter dealing with the pitch envelope can be combined with this and together they form a most useful aid.

Chapter seven looks at various sound effects: zaps, zings, ricochet, space ship etc. There is a small section on sound effects in utility programs, and a small rhythm-generator program.

The next part of the book looks at the Beeb as a musical keyboard and the limitations imposed by this. The last part of this chapter contains a set of programs enabling a bass sequence to be set up with improvisation above it being possible. With the development of editing facilities these could, as the last page suggests, be the basis of a miniature recording studio, albeit extremely limited. Chapter nine contains several programs to produce renditions of Mozart, Tchaikovsky, and Sousa, (Monty Python's signature tune).

Chapters ten and eleven look at the computer as a composer, within the various traditions and rules governing harmony, tonality, etc. Some of the results are I feel rather artificial, because the art of composition is linked with our emotions, experiences and expectations, attributes impossible to program into a computer.

Chapter twelve looks at harmony and transposition, and contains a program for converting from one key to another. This could perhaps be useful for those learning about theory work, or involved in song-arranging.

Generally the programs in the book perform well, and are quite enjoyable to use, the exception being "Hercules". This produces Bizet's 'March of the Torreadors' accompanied by a Hercules strong man rippling his muscles in synchronisation, and with applause to start and finish. This has to be seen to be believed, and can only inspire readers to attempt something slightly more adventurous.

Using Sound and Speech on the BBC Microcomputer by Martin Phillips published by Macmillan at £6.95.

This is a slightly thinner book, and with speech being covered as well, I didn't expect there to be much of interest in the music section, but I was very pleasantly surprised.

The first chapter introduces us to the possibilities of sound on the computer, and the very first program impressed me with the sound effects of waves and seagulls. A second program converts the keyboard to generate four octaves of notes, accompanied by a clear screen display.

Chapter two explains what sound is, and this is more technical than Ian Waugh's book with several diagrams illustrating basic scientific ideas. In this chapter is one of the nicest little programs I've seen for drawing waveforms and their harmonics.

The next two chapters deal with the SOUND and ENVELOPE commands in great detail, with a plethora of small examples to illustrate the text. Obviously a lot of the information is duplicated between the two books, but I did find the Phillips book to be more helpful, mainly because of all the examples. There is a good envelope designer, but the Waugh program wins here because of its simplicity of use and the graphics. Phillips includes a program later in the book, in appendix 3, to complement the one in chapter 4, that will produce graphics, and a print-out of the parameters.

The fifth chapter starts looking at producing music, and there is an interesting section on actually programming musical notation, (used in a later program), to easily reproduce the right symbols.

Chapter six deals with the Acorn speech synthesizer module, and at last it is put into simple layman's language, with lots of examples so I can actually get words out, and hints on compounding syllables together to produce new words.

The appendices are very comprehensive, giving details of the sound buffers, FX calls, speech vocabularies, pitch parameters and two more envelope generators.

In conclusion I found this book to be extremely useful, detailed, with lots of examples and ideas, and I have to say that I preferred it to the Ian Waugh book. Having said this, there are programs in both that I will use, but it is the Phillips book that will be placed in the reference section!

[The subject of music is, of course, very much a matter of personal taste and opinion. We have a high regard for Ian Waugh's book and we expect to start a series by Ian in BEEBUG shortly to assist members interested in making more use of the musical qualities of their Beeb. Ed.]

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

HARD SPACES IN VIEW

If you are preparing a document using View, and insert text with the justification turned on, then View occasionally adds a few spaces, and these can confuse the formatting of the text. To solve this, edit the text with only formatting switched on, and add the justification at a later date. This can be done globally with the FORMAT command, but note that you should not use FORMAT directly after a SCREEN, SHEETS or PRINT commands (this is a known bug in View 1.4 and 2.1); press Escape twice to reset View.

NOISE FOR FIREWORKS - M. Scott

If you liked the fireworks program published in BEEBUG Vol.3 No.5, then you might like to add the following line for some sound effects whilst the program is running.

```
1015 IF RND(10)<4 SOUND 0,-15,4,1
```

Tested on Basic I & II and O.S. 1·2 32k

BUILD YOUR OWN GRAPHICS TABLET (Part 2)

by Ben Miller-Smith

This month sees the second and concluding part of our hardware project designed to help you build your own graphics tablet. This second part is presented with a program to make full use of the tablet.

Having constructed the tablet, the next step is to calibrate it so that the voltages input from the potentiometers to the analogue port can be converted to a corresponding X,Y position on the screen. We need to use the program listed last month (and reproduced again here for convenience).

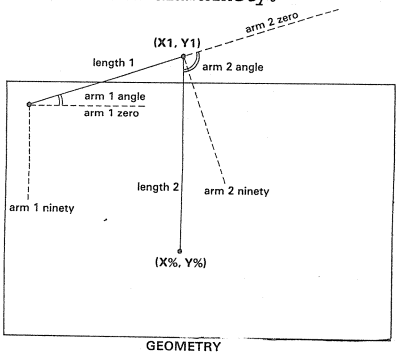
```
10 MODE 7
20 REPEAT
30 PRINT TAB(0,5);SPC(40)
40 PRINT TAB(0,5);ADVAL(1)
50 PRINT TAB(20,5);ADVAL(2)
60 TIME=0:REPEAT UNTIL TIME>50
70 UNTIL FALSE
```

CONVERTING THE VOLTAGES TO ANGLES

Run the above program and position the first arm horizontally along the base board from left to right and note the ADVAL(1) reading (call it arm1zero, say). Swing the arm to a straight down position, at 90 degrees to the first position, and again note the reading (call it arm1ninety). The difference arm1ninety minus arm1zero represents 90 degrees, and arm1zero is the reference ('zero') point, so the angle in any position is given by the expression:

```
(ADVAL(1) - arm1zero) * 90
 / (arm1ninety - arm1zero)
```

with the appropriate values substituted for am1zero and arm1ninety.



GEOMETRY

The second arm is calibrated in a similar fashion, but note that the angle is with respect to the first arm, not the base board. Stretch both arms out horizontally and note the ADVAL(2) reading (arm2zero). Without allowing arm 1 to move, swing arm 2 down to a vertical posiion at 90 degrees to arm 1 and note reading arm2ninety. As above, the arm angle will be given by the same expression, with the appropriate values for arm2zero, arm2ninety and using ADVAL(2) as the input reading. Check that all is well by amending the test program as follows:

Add lines:

```
39 arm1angle = (ADVAL(1) - arm1zero) *
   90 / (arm1ninety - arm1zero)
49 arm2angle = (ADVAL(2) - arm2zero) *
   90 / (arm2ninety - arm2zero)
```

where the armzero and armninety values are those noted during the initial manual checks and calibrations above.

Change lines:

```
40 PRINT TAB(0,5); arm1angle
50 PRINT TAB(20,5); arm2angle
```

Run the program and check that the displayed angles are consistent with the arm positions as you move them about. Note that the angles may be displayed as negative for some typical arm positions.

CALCULATING AND DISPLAYING THE POINTER POSITION

Consider arm 1. Its angle with respect to the base board horizontal is known, so the end point (i.e. the hinge with arm 2) can be calculated if the length is known. Call the length of arm 1 'length1'. The X,Y position of the hinge is then given by:

```
X1 = length1 * COS(RAD(arm1angle))
Y1 = length1 * SIN(RAD(arm1angle))
```

remembering to convert the angle from degrees to radians with the RAD function.

With the position X1,Y1 known, the pointer position at the far end of arm 2 can be calculated from the angle and length of arm 2. Note that the arm2angle value is with respect to arm 1 so we have to allow for this by adding the two angles. We also want the pointer position (PX,PY) with respect to the base board fixed point (the first potentiometer) rather than X1,Y1. The pointer position is thus given by:

```
PX = X1 + length2 * COS(RAD(arm1angle
   + arm2angle))
PY = Y1 + length2 * SIN(RAD(arm1angle
   + arm2angle))
```

The final points concern the transformation of PX and PY into useful graphics coordinates for display. Assuming the arms are of equal length, initialise length1 and length2 to 1023 at the start of the program (if they are unequal, set the long one to 1023, and amend the value of the other to be in the same ratio as the actual lengths). Because of the way the angles are measured, the arms are configured and the position of potentiometer one, the actual screen X%,Y% value for the pointer is given by:

```
X% = INT(PX)
Y% = INT(length2 - PY)
```

GRAPHICS TABLET PROGRAM

Combining all of the above leads to the following program that provides a basic Graphics Tablet input facility, including automatic calibration before use. PROCinit goes through the process of calibrating the zero and ninety degree arm positions with your help, and PROCgetXY then returns the current X,Y position of the pointer in the variables X% and Y%. PROCinit can be simplified if you wish (with some loss of day to day accuracy) by equating 'calpoint(1 to 4)' with the numeric values of 'arm1zero', 'arm1ninety', 'arm2zero' and 'arm2ninety' noted in the manual calibration exercise, and deleting PROCcalibrate and the calls to it, thus avoiding the slightly tedious setting up operation. The main program loop runs forever, and provides three basic commands:

C Clear Screen (& Pen Up)
U Pen Up
D Pen Down

with a cross-hair cursor always showing the Pen position (if on screen). It may easily be modified and extended, for example by using the keyboard to provide additional commands for colour switching, coordinate capture, scaling, etc.

FINISHING TOUCHES

The base board can be provided with small feet so that it can sit on a table top, providing clearance for the body of the board's potentiometer. The feet can be made from small scrap blocks of wood, glued on. If you make your own feet it may be useful to arrange that the board slopes slightly towards you by adjusting the height appropriately.

The potentiometer shafts may be cut shorter, almost flush with the top surfaces of the arms, and small circular paper or cardboard discs glued onto the ends of the shafts. If these discs are marked with calibration points corresponding to the positions of the arms any subsequent slippage of the arm/shaft joints will be obvious, and the joint can be repaired and the unit re-calibrated if necessary.

The base board surface may be marked out with horizontal and vertical lines, especially those showing the corresponding 'screen window', and with reference points for placing A4 and other standard paper sizes on the board. It is also desirable to provide horizontal and vertical 'calibration' lines for the two arms, for use during the initial auto-calibration operation (PROCinit).

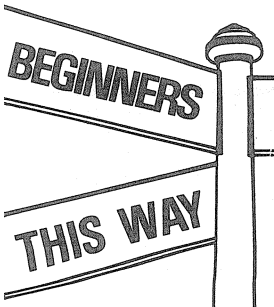
```
10 REM Program GRAPHT
20 REM Version B0.2
30 REM Author  J.B. Miller-Smith
40 REM BEEBUG  DECEMBER 1984
50 REM Program subject to Copyright
60 :
100 ON ERROR GOTO 1770
110 MODE1
120 PROCinit
130 REPEAT
140 key=INKEY(0)
```

```
  150 IF key=ASC(" ") THEN PROCreset:GO
TO190
  160 IF key=ASC("U") THEN pen=4
  170 IF key=ASC("D") THEN pen=5
  180 :
  190 PROCgetXY
  200 PROCcursor(X%,Y%)
  210 PLOT pen,X%,Y%
  220 UNTIL FALSE
  230 END
  240 :
 1000 DEF PROCinit
 1010 :
 1020 REM Limit ADC to 2 Channels
 1030 *FX16,2
 1040 :
 1050 REM Change 'length1%' and/or
 1060 REM 'length2%' to suit hardware
 1070 length1%=1023
 1080 length2%=1023
 1090 :
 1100 REM Cursor off, set colours, &
 1110 REM DIM calibration points array
 1120 VDU 23,1,0;0;0;0;
 1130 VDU19,1,7,0,0,0,19,2,3,0,0,0
 1140 DIM calpoint(4)
 1150 :
 1160 REM For both 0 & 90 deg. positions
 1170 REM on each arm, average the ADVAL
 1180 REM value over 10 readings to help
 1190 REM remove jitter. Prompt user to
 1200 REM position arms as required.
 1210 PROCcalibrate("1 horizontal along
board",1,1)
 1220 PROCcalibrate("1 vertical down bo
ard",2,1)
 1230 PROCcalibrate("2 in line with Arm
 1",3,2)
 1240 PROCcalibrate("2 at right angles
to Arm 1",4,2)
 1250 :
 1260 PROCreset
 1270 ENDPROC
 1280 :
 1290 DEF PROCcalibrate(prompt$,posn,ch
an)
 1300 PRINT'"Make arm ";prompt$;"."'"Pr
ess any key, wait for Beep."'
 1310 REPEAT UNTIL GET
 1320 calpoint(posn)=0
 1330 FOR I%=1 TO 10
 1340 calpoint(posn)=calpoint(posn)+ADV
AL(chan)
 1350 NEXT I%
 1360 VDU7
 1370 calpoint(posn)=calpoint(posn)/10
 1380 ENDPROC
 1390 :
 1400 DEF PROCgetXY
 1410 :
 1420 REM See text for explanations
 1430 armlangle=(ADVAL(1)-calpoint(1))*
90/(calpoint(2)-calpoint(1))
 1440 arm2angle=(ADVAL(2)-calpoint(3))*
90/(calpoint(4)-calpoint(3))
 1450 armlangle=RAD(armlangle)
 1460 arm2angle=RAD(arm2angle)
 1470 X%=INT(length1%*COS(armlangle)+le
ngth2%*COS(armlangle+arm2angle))
 1480  Y%=length2%-INT(length1%*SIN(arm
langle)+length2%*SIN(armlangle+arm2angl
e))
 1490 ENDPROC
 1500 :
 1510 DEF PROCcursor(cx%,cy%)
 1520 MOVE cx%,cy%
 1530 GCOL1,2
 1540 PROCdrawcross
 1550 MOVE oldcx%,oldcy%
 1560 GCOL2,1
 1570 PROCdrawcross
 1580 oldcx%=cx%:oldcy%=cy%
 1590 GCOL0,1
 1600 ENDPROC
 1610 :
 1620 DEF PROCdrawcross
 1630 PLOT 0,16,0
 1640 PLOT 1,-32,0
 1650 PLOT 0,16,16
 1660 PLOT 1,0,-32
 1670 PLOT0,0,16
 1680 ENDPROC
 1690 :
 1700 DEF PROCreset
 1710 CLS
 1720 pen=4
 1730 oldcx%=2000
 1740 oldcy%=2000
 1750 ENDPROC
 1760 :
 1770 ON ERROR OFF
 1780 MODE 7
 1790 IF ERR<>17 REPORT:PRINT " at ";ERL
 1800 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

BETTER LOADING WITH TENSAI TAPE RECORDERS - R. Barnes

If you have a cheap Tensai tape recorder and are having problems getting tape files to load, then try soldering a 0.1 microfarads capacitor in series with a 47K ohm resistor across the 'remote' jack socket.



DEBUGGING PROGRAMS (PART 2)

by Ben Miller-Smith

Ben Miller-Smith continues with the problems of debugging programs by looking this month at run-time errors, their detection and correction.

1. RUN-TIME ERRORS

Run-time errors are those faults which occur during the execution of a program, but which are not due to the Basic interpreter failing to understand and execute the statements making up the program, except perhaps in special circumstances.

Run-time errors may cause all kinds of problems and display all sorts of symptoms as you try and run the program, and a different kind of detective work may be needed to track them down, find the cause, and rectify it. Let's look first at some run-time problems that can still be due to typing or similar mistakes, and which are related to the 'No such variable' type of fatal error discussed last month.

```
360 ...
370 ...
380 droprat = startrate + gravity *
time
390 ...
400 newheight = time * droprate
410 ...
```

This example from the last section was used to illustrate the effect of a typing error in one line (380) causing a fault to be reported in another line (400). The situation would be different if the variable 'droprate' (spelt correctly) had already been used or initialised earlier in the program, around line 200 say, as part of another subtask. In this case line 400 will execute without reporting any error, but will not be using the expected current value of 'droprate' that was supposed to be calculated in line 380 - it will still be the old (line 200) value: no errors will be reported, but the program will not operate correctly.

Other run-time errors can be due to attempts by the program to work out impossible arithmetic or other results, usually because the actual value of one or more variables used in the program has taken on an unexpected value. Examples include:

Division by zero attempted in an expression.

Square root of a negative number attempted.

Attempting to evaluate the LOG of a negative number.

Exceeding the EXP range (up to 88 is valid).

You may quickly find that the commonest cause of such errors is the data input into a program by you or other users. It is surprisingly easy to 'crash' many programs by entering unlikely numbers when some input is requested. This reflects the fact that most programmers are somewhat careless (or optimistic) in their assumptions about the validity of data that will be given to their programs by users. Children especially are likely to enter all sorts of funny data, consciously or unconsciously, but even adults can make typing errors when entering numbers or other information. A good quality program will trap such potential errors and report them to the user, and give him another chance to re-enter the data before continuing with the program - nothing is more annoying than to have a program crash after several minutes of use just because of a typing error, especially if you have spent that time carefully typing in data.

Some run-time software bugs can be due to the incorrect use of VDU instructions, either through a misunderstanding of the required format, or through a typing error. A VDU command in the range VDU2Ø to VDU29 that has accidentally been entered with the second digit missing (thus creating a VDU2) will cause the program to try outputting text to a printer, which if not present or enabled, will cause the program to hang up once it has filled the printer buffer (and probably output some text to the screen). Pressing Escape at this point may give you a clue as to the location of the faulty VDU statement, but not if the incorrect VDU command is part of an initialisation procedure (defining some characters with a VDU23 for example). There is little for it but to look carefully through all the lines containing VDU statements, and check them carefully.

Strange effects can arise from the incorrect format of a VDU instruction, either because of one or more missing parameters or an incorrect parameter separator (some VDU instructions separate parameters by a comma (,), others by a semicolon (;) - always check this carefully). For example:

```
1Ø VDU 19,1,1,Ø
2Ø PRINT "VDU Format ? "
3Ø END
```

will result in the display of "U Format ?", which is a trifle unexpected. This happens because the VDU instruction at line 2Ø is missing two parameters, and thus the next two characters (the 'VD' of line 2Ø) to be output to the screen are taken as the two extra parameters required.

2. TRACING A PROGRAM

Sometimes a bug in a program defies all efforts to find it by the normal methods as outlined above, and a more drastic approach becomes necessary. In these cases it is usually a quite subtle error that is causing the problem, and it may only occur infrequently, and give you little information as to where or why. If you can find some particular combination of inputs or other circumstances that always reproduces the errors then that is half the battle, but this is not always easy. It becomes necessary to Trace the operation of the program, either in whole or in part, using one or both of the following techniques.

The BBC Basic command TRACE ON will cause every line number executed to be printed on the screen in square brackets during the execution of the program. This makes a real mess of the screen, especially if the program is also outputting text or graphics as it runs (but see the TRACER program in this issue for a better solution), but does at least enable you to follow the sequence of events as the program is executing. This may well give you a clue to the problem if you can spot an unexpected departure from the expected sequence, especially if you can create the troublesome error condition with the relevant inputs or whatever. As a first line of attack, proceed as follows.

Add a VDU14 statement to the program as early as possible, but after any mode changes. This switches on the 'paged' mode of operation so that no more than a screenful of information is printed at any one time, and the program and trace output will remain on the screen until you press the Shift key - you thus have time to study the display. Type TRACE ON, and run the program.

If you have a printed listing of the program then you should be able to follow the course of the program's execution in as much detail as necessary, otherwise make mental or written notes of anything that strikes you as odd, especially if the elusive fault occurs (in which case make extra careful notes of the line numbers executed leading up to that point). Go back to a listing of the program and follow the course of execution indicated by the trace listing of executed line numbers. Look especially for conditional or similar tests (IF-THEN etc.) which do not behave as expected, or for loops (FOR-NEXT or REPEAT-UNTIL) that do not terminate correctly.

This process may be sufficient in itself to identify the problem and give

you a flash of insight as to the necessary cure, but in some cases you may have to use the trace facility more selectively, perhaps coupled with some additional information about the run time value of one or more variables. This can be done by including TRACE ON and TRACE OFF statements at selected points in the program to switch the trace listing on and off and, if necessary, by including additional statements in the program that enable you to examine the value of one or more variables in the program at strategic points. The simplest way of doing this is often to include extra lines of the form:

```
135 IF droprate > 100 THEN PRINT
"droprate =";droprate : REPEAT UNTIL
GET
```

if you suspect that the value of 'droprate' may be being miscalculated at some point, and values over 100 are invalid. On executing this line the current value of 'droprate' will be printed if it exceeds 100, and you then have the option of pressing a key to continue, or pressing Escape. After an Escape you will also have the option of asking for the value of any other variables by using a direct command of the form:

```
PRINT variable1,variable2,variable3
```

which may help to establish what has gone wrong.

The above line of attack can sometimes be improved by including an extra procedure in the program that will display on the screen (in an unused corner say) the value of one or more variables at strategic points in the program. An example of such a procedure could be:

```
9000 DEFPROCvartrace(v1,v2,v3,posx%,
posy%)
9010 LOCAL cursx%,cursy%
9020 cursx%=POS: cursy%=VPOS
9030 PRINTTAB(posx%,posy%);v1,v2,v3
9040 REPEAT UNTIL GET
9050 PRINTTAB(cursx%,cursy%);
9060 ENDPROC
```

This procedure stores the current position of the text cursor in 'cursx%' and 'cursy%', and prints the value of the three variables named in the procedure call at position 'posx%' and 'posy%' also given in the call. It then awaits any key depression (giving you time to note the variable values), restores the original text cursor position, and exits.

Calls to this procedure can be scattered around in the program to check on the current value of relevant variables - a typical call could be:

```
120 PROCvartrace(droprate,gravity,0,
5,20)
```

which would print the current values of 'droprate' and 'gravity' (and a zero, to make up the third parameter) at text cursor position (5,20). Note that you can make calls to this procedure conditional on the value of other variables or situations in the program by including the call in an IF statement if necessary.

3. CONCLUSION

Fault finding in BBC Basic programs is usually not too difficult, thanks to the powerful and 'user-friendly' error handling software built into the system, but there are occasions in which the reported fault is misleading, and some detective work may be necessary. The more practice you have in debugging programs, the quicker you will recognise the different types of error that may occur. There is a lot to be said for deliberately introducing errors into a known, working program (better still, get someone else to do it), and experimenting with the debugging techniques covered in these notes. Bugs can be elusive, but they are there to be found. Happy Hunting!

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MACROS IN ASSEMBLER

If you are using functions to generate macros in the Beeb's 6502 assembler, you will find it more reliable to use EQUS to call your code (and return a null string) than to use OPT (and return the current 'pass'). The reason for this is that there are some obscure bugs in the OPT section of Basic which don't occur when using EQUS.

Tested on Basic I & II and O.S. 1.2 32k

GEORGE AND THE DRAGON

by O.R. Thomas

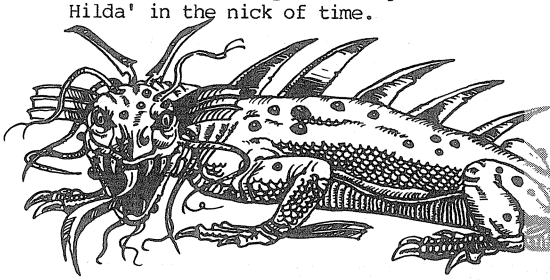
Now it's time for some light-hearted relief from zapping and blasting aliens. No sooner has your Christmas pudding settled than you are off to rescue 'Hideous Hilda' from the flames of the dragon, in this fast and exciting action game from O.R. Thomas.

The idea of the game is to run along each level, jumping over the moving hole to collect the key which will allow George to climb the ladder up to the next level. George needs to reach the top level to free 'Hideous Hilda' before the dragon's flame reaches her. If you succeed in freeing Hilda, you proceed onto the next screen, where in addition to moving holes in the floor, you will have to face the additional hazard of arrows being fired just above your head.

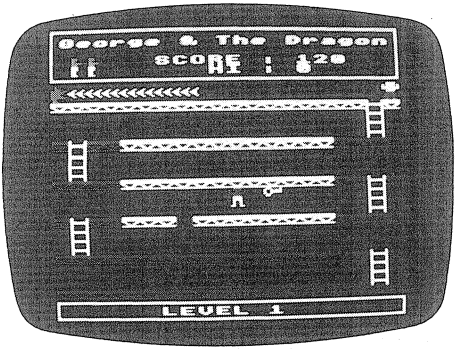
The keys to use to play this game are 'Z' and 'X' for left and right, plus '/' to climb the ladders and 'Shift' to jump. There are nine different skill levels ranging from 1, which is fast, to 9 which is slow. As you complete each screen, the game automatically gets faster and faster.

Remember to take extra care when typing in the character and string definitions, as mistakes here will corrupt the screen display when you run the program.

So now you are ready to undertake your perilous quest, to thwart the wicked dragon by reaching 'Hideous Hilda' in the nick of time.

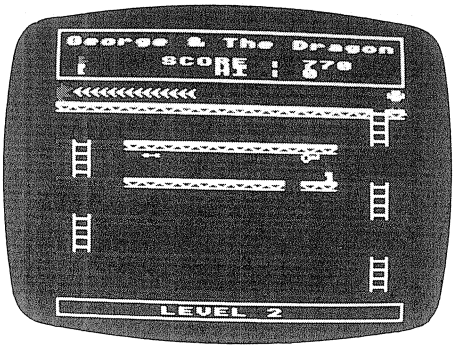


```
10 REM PROGRAM GEORGE
20 REM VERSION B0.2
30 REM AUTHOR  O.R.Thomas
40 REM BEEBUG  DECEMBER 1984
50 REM PROGRAM SUBJECT TO COPYRIGHT
60 :
100 ON ERROR GOTO 2950
110 MODE7
120 PROCinstructions
130 hi%=0:MODE 7:REPEAT
140 PROCskill:PROCvariables
150 FOR lives%=2 TO 0 STEP-1
160 MODE5:PROCcharacters
170 PROCscreen:PROCsetup
180 time%=0:totaltime%=0
190 REPEAT:time%=time%+1
200 IF time%>=pause% time%=0:totaltim
e%=totaltime%+1:VDU5:GCOL0,2:MOVE(total
time%*32)+32,816:PRINTflame$:VDU4
210 IF totaltime%=37 dead%=TRUE
220 PROCman
230 IF NOT jump% AND xpos%=holepos% d
ead%=TRUE:GOTO290
240 PROChole
250 IF NOT jump% AND xpos%=holepos% d
ead%=TRUE:GOTO290
260 IF xpos%=keypos% AND ypos%=keyhei
ght% key%=TRUE:SOUND1,-15,100,1:score%=
score%+50:PROCscore:keypos%=20
270 IF arrow% AND xpos%=arrowpos% AND
 ypos%=keyheight% dead%=TRUE:GOTO290
280 IF arrow% PROCarrow
290 UNTIL dead%:PROCdeath
300 FOR ag%=1 TO 15:SOUND0,-15+ag%,10
0,2:NEXT
310 jump%=FALSE:jumpno%=0:ychange%=0:
xchange%=0
320 dead%=FALSE:PROCscore:NEXT
330 COLOUR2:PRINTTAB(4,17)STRING$(12,
CHR$32);TAB(4,18)" GAME  OVER ";TAB(4,1
9)STRING$(12,CHR$32)
340 TIME=0:REPEAT UNTIL TIME>300
350 IF score%>hi% hi%=score%
360 PROCscodisp
370 UNTIL Q$="N":MODE 7
380 END
390 :
1000 DEF PROCvariables
1010 key%=FALSE:dead%=FALSE:score%=0
1020 level%=1:jump%=FALSE:jumpno%=0
1030 arrow%=FALSE:arrowpos%=10
1040 ENDPROC
1050 :
```




```
1060 DEF PROCcharacters
1070 VDU23,224,56,56,56,48,62,62,56,56
:VDU23,225,56,56,56,48,48,48,48,56
1080 VDU23,226,28,28,28,12,124,124,28,
28:VDU23,227,28,28,28,12,12,12,12,28
1090 VDU23,228,60,60,189,153,255,255,6
0,60:VDU23,229,60,60,60,36,36,36,36,102
1100 VDU23,230,0,0,0,0,24,44,118,175
1110 VDU23,231,32,32,48,60,251,255,254
,252:VDU23,232,248,252,255,254,224,96,1
12,80
1120 VDU23,233,0,16,32,96,224,96,32,16
:VDU23,234,60,60,126,126,255,255,36,102
1130 VDU23,235,96,97,149,159,159,144,9
6,96:VDU23,236,255,255,255,129,66,36,24
,255
1140 VDU23,237,129,129,129,255,255,129
,129,129:VDU23,238,0,0,0,99,254,99,0,0
1150 VDU23,239,0,0,0,198,127,198,0,0:VD
U23,240,129,129,129,129,129,129,129,129
1160 VDU19,3,4;0;
1170 VDU23;8202;0;0;0;
1180 right$=CHR$17+CHR$1+CHR$224+CHR$1
0+CHR$8+CHR$17+CHR$2+CHR$225
1190 left$=CHR$17+CHR$1+CHR$226+CHR$10
+CHR$8+CHR$17+CHR$2+CHR$227
1200 still$=CHR$17+CHR$1+CHR$228+CHR$1
0+CHR$8+CHR$17+CHR$2+CHR$229
1210 dead$=CHR$32+CHR$10+CHR$8+CHR$17+
CHR$2+CHR$230
1220 dragon$=CHR$17+CHR$1+CHR$231+CHR$
10+CHR$8+CHR$232
1230 flame$=CHR$17+CHR$2+CHR$233
1240 maiden$=CHR$17+CHR$2+CHR$228+CHR$
10+CHR$8+CHR$17+CHR$1+CHR$234
1250 key$=CHR$17+CHR$2+CHR$235
1260 safefloor$=CHR$17+CHR$1+CHR$236
1270 dangerfloor$=CHR$17+CHR$3+CHR$236
1280 ladder$=CHR$17+CHR$2+CHR$237
1290 doubleladder$=CHR$17+CHR$2+CHR$23
7+CHR$10+CHR$8+CHR$237
1300 arrowleft$=CHR$17+CHR$2+CHR$238
1310 arrowright$=CHR$17+CHR$2+CHR$239
1320 gap$=CHR$17+CHR$1+CHR$240
1330 blank$=CHR$32+CHR$10+CHR$8+CHR$32
1340 ENDPROC
1350 :
1360 DEF PROCscreen
1370 GCOL0,130:VDU24,0;847;1279;1023;
1380 CLG:VDU24,0;16;1279;80;:CLG
1390 GCOL0,128:VDU24,8;855;1271;1015;
1400 CLG:VDU24,8;24;1271;72;:CLG
1410 VDU24,0;0;1279;1023;:VDU5
1420 GCOL0,1:MOVE24,991
1430 PRINT"George & The Dragon"
1440 GCOL0,2:MOVE32,995
1450 PRINT"George & The Dragon"
1460 VDU4:PROCscore
1470 PRINTTAB(6,3)"SCORE :"
1480 PRINTTAB(9,4)"HI :"
1490 PRINTTAB(6,30)"LEVEL ";level%
1500 PRINTTAB(0,6)dragon$
1510 PRINTTAB(19,6)maiden$
1520 PRINTTAB(1,26)still$
1530 xpos%=1:ypos%=26
1540 PRINTTAB(0,8)STRING$(20,dangerflo
or$)
1550 FOR ah%=12 TO 28 STEP 4
1560 PRINTTAB(0,ah%)STRING$(20,safeflo
or$)
1570 PRINTTAB(4,ah%)STRING$(12,dangerf
loor$)
1580 NEXT:anyx%=18
1590 FOR ah%=8 TO 24 STEP 4
1600 FOR anyy%=ah% TO ah%+3
1610 PRINTTAB(anyx%,anyy%)ladder$
1620 NEXT
1630 IF anyx%=18 anyx%=1 ELSE anyx%=18
1640 NEXT
1650 ENDPROC
1660 :
1670 DEF PROCscore
1680 COLOUR2
1690 IF lives%<1 GOTO 1730
1700 FOR ah%=0 TO lives%-1
1710 PRINTTAB(ah%+1,3)right$
1720 NEXT
1730 PRINTTAB(lives%+2,3)blank$
1740 PRINTTAB(14,3);score%
1750 PRINTTAB(14,4);hi%
1760 ENDPROC
1770 :
1780 DEF PROCman
1790 IF jump% PROCjump:GOTO1860
1800 xchange%=0
1810 ychange%=0
1820 IF INKEY(-98) AND NOT jump% xchan
ge%=-1
1830 IF INKEY(-67) AND NOT jump% xchan
ge%=1
1840 IF INKEY(-1) jump%=TRUE
1850 IF INKEY(-105) AND POINT(xpos%*64
,(32-ypos%)*32+36)=2 AND key% PROCclimb
```

→




```
 1860 IF xchange%<>0 OR jump% PRINTTAB(
xpos%,ypos%)blank$:IF POINT(xpos%*64,(3
2-ypos%)*32+36)=2 AND NOT jump% OR POIN
T(xpos%*64,(32-ypos%)*32+4)=2 PRINTTAB(
xpos%,ypos%)doubleladder$
 1870 IF jumpno%=3:jumpno%=0:jump%=FALSE
 1880 xpos%=xpos%+xchange%
 1890 ypos%=ypos%+ychange%
 1900 IF xpos%<0 xpos%=0
 1910 IF xpos%>19 xpos%=19
 1920 IF xchange%=1 PRINTTAB(xpos%,ypos
%)right$
 1930 IF xchange%=0 PRINTTAB(xpos%,ypos
%)still$
 1940 IF xchange%=-1 PRINTTAB(xpos%,ypo
s%)left$
 1950 ENDPROC
 1960 :
 1970 DEF PROCclimb
 1980 key%=FALSE
 1990 FOR ah%=1 TO 4
 2000 SOUND1,-15,50*ah%,2
 2010 IF ah%=1 PRINTTAB(xpos%,ypos%+1)l
adder$:GOTO2030
 2020 PRINTTAB(xpos%,ypos%+1)doubleladd
er$
 2030 ypos%=ypos%-1
 2040 PRINTTAB(xpos%,ypos%)still$
 2050 NEXT
 2060 score%=score%+10:PROCscore
 2070 IF ypos%=6 CLS:level%=level%+1:sc
ore%=score%+((37-totaltime%)*10):PROCsc
reen:totaltime%=0:IF pause%>5 pause%=pa
use%-1
 2080 PRINTTAB(arrowpos%,keyheight%);CH
R$32
 2090 IF POINT(arrowpos%*64,(32-keyheig
ht%)*32+4)=2 PRINTTAB(arrowpos%,keyheig
ht%)ladder$
 2100 PROCsetup
 2110 ENDPROC
 2120 :
 2130 DEF PROCjump
 2140 jumpno%=jumpno%+1
 2150 IF jumpno%=1 ychange%=-1
 2160 IF jumpno%=2 ychange%=0
 2170 IF jumpno%=3 ychange%=1
 2180 SOUND1,-10,140+(30*ychange%),1
 2190 ENDPROC
 2200 :
 2210 DEF PROCsetup
 2220 IF ypos%<26 PRINTTAB(4,ypos%+6)ST
RING$(12,safefloor$)
 2230 holepos%=10:holedir%=1
 2240 keypos%=RND(12)+3:keyheight%=ypos
%-1
 2250 key%=FALSE:PRINTTAB(keypos%,ypos%
-1)key$
 2260 IF level%>1 arrow%=TRUE:arrowpos%
=RND(18):IF arrowpos%<10 dir%=1 ELSE di
r%=-1
 2270 ENDPROC
 2280 :
 2290 DEF PROCarrow
 2300 PRINTTAB(arrowpos%,keyheight%);CH
R$32
 2310 IF arrowpos%=keypos% PRINTTAB(arr
owpos%,keyheight%);key$
 2320 IF POINT(arrowpos%*64,(32-keyheig
ht%)*32+4)=2 PRINTTAB(arrowpos%,keyheig
ht%)ladder$
 2330 arrowpos%=arrowpos%+dir%
 2340 IF dir%=1 AND arrowpos%>19 dir%=-
1:arrowpos%=19
 2350 IF dir%=-1 AND arrowpos%<0 dir%=1
:arrowpos%=0
 2360 IF dir%=1 PRINTTAB(arrowpos%,keyh
eight%)arrowright$
 2370 IF dir%=-1 PRINTTAB(arrowpos%,key
height%)arrowleft$
 2380 ENDPROC
 2390 :
 2400 DEF PROChole
 2410 PRINTTAB(holepos%,keyheight%+3)da
ngerfloor$
 2420 holepos%=holepos%+holedir%
 2430 IF holepos%>15 holepos%=15:holedi
r%=-1
 2440 IF holepos%<4 holepos%=4:holedir%
=1
 2450 PRINTTAB(holepos%,keyheight%+3)ga
p$
 2460 ENDPROC
 2470 :
 2480 DEF PROCdeath
 2490 IF totaltime%=37 PRINTTAB(19,6)de
ad$:GOTO 2550
 2500 REPEAT
 2510 PRINTTAB(xpos%,ypos%)CHR$32
 2520 ypos%=ypos%+1
 2530 PRINTTAB(xpos%,ypos%)still$
 2540 UNTIL POINT(xpos%*64+32,(32-(ypos
%+2))*32-4)>0 OR ypos%=27
 2550 PRINTTAB(xpos%,ypos%)dead$
```

```
2560 ENDPROC
2570 :
2580 DEF PROCinstructions
2590 CLS
2600 VDU23;8202;0;0;0;
2610 PRINT'CHR$129;CHR$157;TAB(8);CHR$
131;CHR$141;"GEORGE AND THE DRAGON"
2620 PRINTCHR$129;CHR$157;TAB(8);CHR$1
31;CHR$141;"GEORGE AND THE DRAGON"
2630 PRINTTAB(13)CHR$134;"by O.R.Thoma
s"'
2640 PRINTCHR$133;" Help George to do
dge arrows and leap"CHR$133;"over hol
es as he rushes to grasp the"CHR$133;
"keys allowing him further up th
e"CHR$133;"battlements. But hurry -
 you must";
2650 PRINTCHR$133;"rescue Hideous
Hilda before the"CHR$133;"dragon's
flames reach her."
2660 PRINT'''TAB(4)CHR$133;"The contro
ls are as follows :"'TAB(13)CHR$131;"Z
 - left"'TAB(13)CHR$131;"X - rig
ht"'TAB(13)CHR$131;"/ - climb"'TAB(
9)CHR$131;"SHIFT - jump"
2670 PRINTTAB(0,24)CHR$129;CHR$157;TAB
(7)CHR$136;CHR$131;"PRESS SPACE TO CONT
INUE";
2680 *FX15,1
2690 REPEAT UNTIL GET=32
2700 ENDPROC
2710 :
2720 DEF PROCskill
2730 VDU23;8202;0;0;0;
2740 PRINTTAB(9,14)CHR$141;CHR$129;"SK
ILL LEVEL (1-9):"
2750 PRINTTAB(9)CHR$141;CHR$131;"SKILL
 LEVEL (1-9):"
2760 *FX15,1
2770 pause%=5+(GET-48)
2780 IF pause%<6 OR pause%>14 GOTO 277
2790 PRINTTAB(29,14);pause%-5
2800 PRINTTAB(29,15);pause%-5
2810 TIME=0:REPEAT UNTIL TIME>70
2820 ENDPROC
2830 DEFPROCscodisp
2840 VDU22,7
2850 FORA%=2TO3
2860 PRINTTAB(9,A%)CHR$141;CHR$(127+A%
);"YOUR SCORE WAS ";STR$(score%)
2870 NEXT
2880 PRINT''':FORA%=1TO2
2890 PRINTTAB(10)CHR$141;CHR$(132+A%);
"ANOTHER GAME ?";:NEXT
2900 *FX15,1
2910 Q$=GET$:PRINTCHR$11;Q$;CHR$8;CHR$
10;Q$
2920 ENDPROC
2930 :
2940 ON ERROR OFF:MODE 7
2950 IF ERR=17 END
2960 REPORT:PRINT" at line ";ERL:END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

COLOURFUL GCOL PARAMETERS - Paul Watts

As you may know, you can get the operating system to draw in two striped colours by using the GCOL statement with silly parameters. Here is a routine (non-Tube compatible as it stands) that allows you to specify the two colours yourself. The displayed colours will, of course, be dependent upon the current chosen mode.

```
DEF PROCgcol(option,first,second) LOCAL a,b
GCOL option,first:a=?&359 AND &AA:GCOL option,second
b=?&359 AND &55:?&359=a + b:ENDPROC
```

READING SIDEWAYS ROMS

If you want to read a byte x from ROM y, then use the function below, with x as the first parameter and y as the second:

```
DEF FNpeek(!&F6,Y%) = USR(&FFB9) AND &FF
```

For example, to read the byte at &89AB in ROM 7, then the following would be used:

```
PRINT FNpeek(&89AB,7)
```

STRANGE VARIABLES - Tony Walsh

If you have Toolkit and you wish to make a program unalterable, then try changing variables to built in words such as PRINT (with the search and replace option), but omit the £ sign to indicate that it is a Basic keyword. Toolkit then replaces your variable for one spelt like the Basic keyword, but not tokenised. Most attempts to change the program will result in the variable being tokenised, and thus Basic producing an error when the program is rui .

Tested on Basic I & II and O.S. 1-2 32k

CHRISTMAS FRUIT MACHINE

by A. Hayden

Have you ever wanted to play a fruit machine without the risk of losing a pocketful of money? Well BEEBUG gives you the chance to do just that with A.Hayden's computerised and colourful version.

Fruit Machine is a colourful (and noisy) game in which the player attempts to increase their wealth by gambling their money. You start the game with £2.00, and each spin of the reels costs 10 pence, with a chance of winning anything from 20 pence to £3.00 (and not just in tokens either).

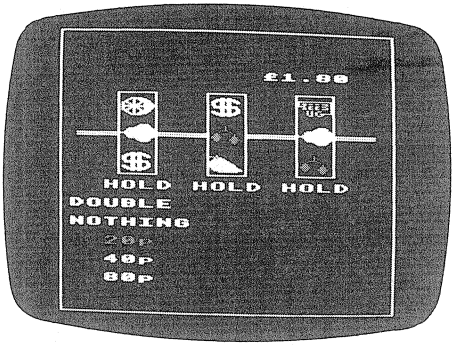
There are three reels with eight different symbols (there are a total of 20 symbols on each reel, but some occur more times than others) and three of these symbols are displayed on each reel at any one time. The middle line of each displayed reel is called the win line, and all but one of the wins has to occur by getting two of the same symbol on the first two reels or by getting all three the same. There is however one exception to this which is the BEEBUG symbol. To win with this, you only need to have the symbols showing, and not just on the win line.

INSTRUCTIONS

To start the reels spinning press the 'S' key. At this point three things may happen; a 'Shuffle', a 'Hold', or the reels just spin. Given the choice, you can hold each reel by pressing the appropriate number (e.g pressing 1 and 3 will hold reels 1 and 3) and pressing C will cancel the holds. If you get a shuffle, then pressing the space bar will shuffle the reels before offering you a hold (this does not cost you any money and will spin the reels around approximately halfway). If you don't wish to shuffle the reels (if you already have a win) then you can press C to cancel the shuffle, and then hold your win.

When you get a win you will be given a chance to gamble your winnings (if they are 20p, 40p or 80p). To gamble, press C (to double your winnings or lose it all) or press S to collect your winnings.

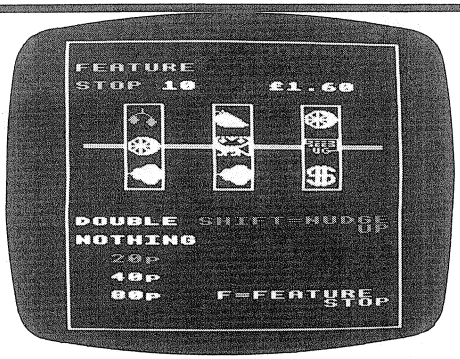




If after the reels have stopped spinning you get a 'Feature Stop', you can then press F which will randomly select a number (displayed on the screen) between 1 and 10. This number represents the number of available 'nudges', and you must then try and nudge the reels to produce a win. To nudge a reel down you simply press the corresponding number key (1, 2 or 3), and to nudge a reel up you simply press Shift and a number key together.

If you are not a regular and compulsive player of fruit machines some of these terms and instructions may sound confusing. As soon as you start playing everything will quickly become clear and you will probably become as addicted as the next person.

```
  10 REM PROGRAM FRUIT
  20 REM VERSION B0.7
  30 REM AUTHOR  A.HAYDEN
  40 REM BEEBUG  DECEMBER 1984
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 ON ERROR GOTO 3830
 110 MODE 7:X=RND(-TIME)
 120 DIM F$(8),X%(3),R$(3),RL$(3),H%(3
),PR%(3)
 130 REPEAT:PROCchars
 140 MODE2:PROCvar:PROCset
 150 REPEAT:PROCroll
 160 IF F%=0 OR DON%=0 THEN PROCcheck
 170 F%=0:DON%=0:UNTIL MN%=0 OR MN%>999
```




```
  180 PROCmore:UNTILØ
  190 END
  200 :
 1000 DEF PROCvar
 1010 RESTORE 3810
 1020 FOR A%=Ø TO 7:READ B%:C%=A%*4+224
 1030 F$(A%)=CHR$17+CHR$(B%)+CHR$17+CHR
$128+CHR$(C%)+CHR$(C%+1)+CHR$8+CHR$8+CH
R$10+CHR$(C%+2)+CHR$(C%+3)
 1040 NEXT
 1050 RL$(1)="AGCRMPCMAOGCPBGCRMAP"
 1060 RL$(2)="MRBAOCGPCAMCGPAGRMCP"
 1070 RL$(3)="CRPMBACMBOAPGRCPCGMA"
 1080 FOR A%=1 TO 3:PR%(A%)=RND(18):R$(
A%)=MID$(RL$(A%),PR%(A%),3):H%(A%)=Ø:NE
XT
 1090 MN%=20:S$="RMGAPOBC"
 1100 PRIZE%=Ø:X=4:F%=Ø:DON%=Ø
 1110 ENDPROC
 1120 :
 1130 DEF PROCroll:*FX15,1
 1140 REPEAT GT$=GET$:UNTILGT$="S":GM%=Ø
 1150 IF RND(8)=1 THEN PROCshuffle:PROC
hold:GOTO1170
 1160 IF RND(X)=1 THEN PROChold
 1170 X=4:X%(1)=RND(10)+20
 1180 Y%=RND(8)+3:X%(2)=X%(1)+Y%
 1190 X%(3)=X%(2)+Y%:MN%=MN%-1
 1200 PROCmoney(MN%)
 1210 FOR R%=1 TO X%(3):FOR S%=1 TO 3
 1220 IF R%=X%(S%) AND NOT H%(S%) THEN
SOUND Ø,-15,4,2
 1230 IF H%(S%) OR R%>=X%(S%) THEN 1310
 1240 PR%(S%)=PR%(S%)-1
 1250 IF PR%(S%)=Ø THEN PR%(S%)=20
 1260 R$(S%)=MID$(RL$(S%),PR%(S%),1)+R$
(S%)
 1270 R$(S%)=LEFT$(R$(S%),3)
 1280 FOR A%=1 TO 3
 1290 PRINTTAB(5*S%-1,A%*3+5);F$(INSTR(
S$,MID$(R$(S%),A%,1))-1)
 1300 NEXT
 1310 NEXT,
 1320 IF RND(15)=15 THEN PROCfstop
 1330 ENDPROC
 1340 :
 1350 DEF PROCmoney(M%)
 1360 COLOUR 6:COLOUR 128
 1370 PRINTTAB(12,5);"£";M% DIV 10;".";
M% MOD 10;"Ø "
 1380 ENDPROC
 1390 :
 1400 DEF PROCmore
 1410 FORA=1TO5000:NEXT
 1420 VDU 22,7
 1430 PRINTTAB(10,10);CHR$133;"Another
go (Y/N)?";
 1440 A=GET
 1450 IF A=89 THEN ENDPROC
 1460 IF A<>78 THEN 1420
 1470 PRINT:END
 1480 :
 1490 DEF PROCcheck:PRIZE%=Ø
 1500 H%(1)=Ø:H%(2)=Ø:H%(3)=Ø:GM%=Ø
 1510 R1$=MID$(R$(1),2,1)
 1520 R2$=MID$(R$(2),2,1)
 1530 R3$=MID$(R$(3),2,1)
 1540 IF INSTR(R$(1),"B")>Ø AND INSTR(R
$(2),"B")>Ø AND INSTR(R$(3),"B")>Ø THEN
 PROC3beebugs:GM%=1:GOTO1580
 1550 IF INSTR(R$(1),"B")>Ø AND INSTR(R
$(2),"B")>Ø THEN PROC2win:GM%=1:GOTO1580
 1560 IF R1$=R2$ AND R1$=R3$ THEN PROC3
win:GM%=1:GOTO1580
 1570 IF R1$=R2$ THEN PROC2win
 1580 PROCmoney(MN%):PROCdispmon
 1590 ENDPROC
 1600 :
 1610 DEF PROCdispmon
 1620 COLOUR3:PRINTTAB(1,19)"DOUBLE";TA
B(1,21)"NOTHING":FORC=1TO4
 1630 COLOURC:PRINTTAB(3,21+(2*C));(2^C
)*10;"p":NEXT
 1640 ENDPROC
 1650 :
 1660 DEF PROC2win:IF GM%=1 ENDPROC
 1670 PRIZE%=FNdorn(2):MN%=MN%+PRIZE%:G
M%=1
 1680 IF PRIZE%=Ø THEN SOUND 3,-15,30,1
Ø:GOTO1730
 1690 SOUND 1,-15,100,10
 1700 SOUND 1,-15,88,5
 1710 SOUND 1,-15,104,5
 1720 SOUND 1,-15,96,10
 1730 X=2:ENDPROC
 1740 :
 1750 DEF PROC3beebugs:IF GM%=1 ENDPROC
 1760 MN%=MN%+10:SOUND 1,-15,88,5
 1770 SOUND 1,-15,92,5
 1780 SOUND 1,-15,112,10
 1790 SOUND 1,-15,108,5
 1800 SOUND 1,-15,120,10
 1810 X=3:GM%=1:ENDPROC
 1820 :
 1830 DEF PROCset:COLOUR 128
 1840 CLS
 1850 COLOUR 1
```



```
 1860 GCOL0,3:MOVE100,632:DRAW1180,632
 1870 MOVE100,640:DRAW1180,640
 1880 COLOUR 4
 1890 PRINTTAB(12,1);"RUNNING"
 1900 PRINTTAB(13,3);"TOTAL"
 1910 COLOUR 6
 1920 PROCmoney(MN%)
 1930 PROCdispmon
 1940 FOR A=8 TO 15
 1950 FOR C=4 TO 14 STEP 5
 1960 PRINTTAB(C,A);"  "
 1970 NEXT,
 1980 FOR A%=1 TO 3
 1990 FOR B%=1 TO 3
 2000 G$=MID$(R$(A%),B%,1)
 2010 PRINTTAB(5*A%-1,5+3*B%);F$(INSTR(
S$,G$)-1)
 2020 NEXT,
 2030 GCOL0,6:MOVE32,16:DRAW32,1008:DRA
W1248,1008:DRAW1248,16:DRAW32,16
 2040 FOR X%=0 TO2:MOVE 250+(X%*320),500
 2050 DRAW 390+(X%*320),500:DRAW 390+(X
%*320),780:DRAW 250+(X%*320),780:DRAW 2
50+(X%*320),500
 2060 NEXT:VDU23,1,0;0;0;0;
 2070 ENDPROC
 2080 :
 2090 DEF PROChold:H%(1)=0:H%(2)=0:H%(3
)=0
 2100 COLOUR4:PRINTTAB(8,25);"1/2/3=HOL
DS":*FX15 1
 2110 COLOUR 14:COLOUR 128
 2120 PRINTTAB(3,17);"HOLD HOLD HOLD"
 2130 REPEAT:A=GET-48
 2140 UNTIL (A>0 AND A<4) OR A=35 OR A=
19
 2150 IF A=19 THEN FORA2=1TO3:H%(A2)=0:
NEXT:GOTO 2110
 2160 IF A=35 THEN 2210
 2170 H%(A)=TRUE
 2180 COLOUR 4
 2190 PRINTTAB(5*A-2,17);"HOLD"
 2200 GOTO 2130
 2210 COLOUR 0
 2220 FOR A%=1 TO 3
 2230 PRINTTAB(5*A%-2,17);"HOLD"
 2240 NEXT:PRINTTAB(8,25);SPC(11)
 2250 ENDPROC
 2260 :
 2270 DEF PROC3win:IF GM%=1 ENDPROC
 2280 IF R1$="C" OR R1$="P" OR R1$="A"
THEN PRIZE%=4
 2290 IF R1$="M" OR R1$="G" THEN PRIZE%
=8
 2300 IF R1$="R" THEN PRIZE%=15
 2310 IF R1$="O" THEN PRIZE%=20:PROCtune
 2320 IF PRIZE%<10 THEN PRIZE%=FNdorn(P
RIZE%)
 2330 IF PRIZE%<20 AND PRIZE%>0 THEN SO
UND 1,-15,109,5:SOUND 1,-15,89,5:SOUND1
,-15,97,5
 2340 MN%=MN%+PRIZE%
 2350 PROCmoney(MN%)
 2360 GM%=1:ENDPROC
 2370 :
 2380 DEF PROCtune
 2390 RESTORE 2630
 2400 READ L%
 2410 FOR Y%=1 TO L%
 2420 READ P,D
 2430 SOUND 1,-15,P,D:IF P=0 P=-1
 2440 SOUND 2,-15,P+1,D
 2450 NEXT
 2460 X=3:ENDPROC
 2470 :
 2480 DEFFNdorn(U%)
 2490 J%=1:COLOUR 128:PROCup(U%)
 2500 REPEAT:SOUND 2,-15,53,3
 2510 COLOUR 4
 2520 PRINTTAB(1,19);"DOUBLE"
 2530 A=INKEY(15):IF A=67 THEN U%=U%*2:
PROCup(U%)
 2540 IF A=83 THEN J%=0:GOTO2610
 2550 COLOUR 3:PRINTTAB(1,19);"DOUBLE"
 2560 SOUND 2,-15,61,3
 2570 COLOUR 4:PRINTTAB(1,21);"NOTHING"
 2580 A=INKEY(15):IF A=67 THEN U%=0:SOU
ND 1,-15,53,5
 2590 IF A=83 THEN J%=0
 2600 COLOUR 3:PRINTTAB(1,21);"NOTHING"
 2610 UNTIL U%=16 OR U%=0 OR J%=0
 2620 DON%=1:=U%
 2630 DATA 16,117,7,145,12,117,3,120,3
 2640 DATA 123,3,126,3,129,10,0,2,80,7
 2650 DATA 95,12,80,3,77,3,74,3,71,3
 2660 DATA 68,3,65,10
 2670 :
 2680 DEF PROCup(US%)
 2690 COLOUR7
 2700 IF US%=2 THEN PRINTTAB(3,23);"20p"
 2710 IF US%=4 THEN PRINTTAB(3,25);"40p
";TAB(3,23);"20p"
 2720 IF US%=8 THEN PRINTTAB(3,27);"80p
";TAB(3,25);"40p"
 2730 IF US%=16 THEN PRINTTAB(3,29);"16
0p":TI=TIME:REPEATUNTILTIME>100+TI
 2740 ENDPROC
 2750 :
 2760 DEF PROCshuffle:*FX 15,1
 2770 COLOUR 2:PRINTTAB(1,1);"SHUFFLE";
TAB(8,30)""" """=SHUFFLE"
 2780 L%=50:REPEAT:L%=L%+4:S=INKEY(0):S
OUND 1,-15,L%,2
 2790 UNTIL S=32 OR S=67
 2800 IF S=32 THEN PROCshuf
 2810 PRINTTAB(1,1);SPC(7);TAB(8,30);SP
C(11)
 2820 ENDPROC
 2830 :
 2840 DEF PROCshuf:X%(1)=-(RND(8)+2)
 2850 X%(2)=RND(8)+2
```



→

```
 2860 X%(3)=-(RND(8)+2)
 2870 FOR Y%=1 TO 10
 2880 FOR S%=1 TO 3
 2890 IF ABS(X%(S%))=Y% THEN 2980
 2900 PR%(S%)=PR%(S%)+SGN(X%(S%))
 2910 IF PR%(S%)=22 THEN PR%(S%)=1
 2920 IF PR%(S%)=0 THEN PR%(S%)=21
 2930 IF X%(S%)<0 THEN R$(S%)=LEFT$((MI
D$(RL$(S%),PR%(S%),1)+R$(S%)),3)
 2940 IF X%(S%)>0 THEN R$(S%)=RIGHT$((R
$(S%)+MID$(RL$(S%),PR%(S%),1)),3)
 2950 FOR A%=1 TO 3
 2960 PRINTTAB(5*S%-1,A%*3+5);F$(INSTR(
S$,MID$(R$(S%),A%,1))-1)
 2970 NEXT
 2980 NEXT,
 2990 PROCcheck
 3000 ENDPROC
 3010 :
 3020 DEF PROCfstop
 3030 *FX15,1
 3040 COLOUR 1:X=1
 3050 COLOUR 128
 3060 PRINTTAB(1,3);"FEATURE";TAB(1,5);
"STOP";TAB(8,19);"SHIFT=NUDGE";TAB(17,2
0)"UP":COLOUR5:PRINTTAB(9,27);"F=FEATUR
E";TAB(15,28);"STOP"
 3070 PROCstop
 3080 PRINTTAB(1,3);SPC(7);TAB(1,5);SPC
(4);TAB(8,19);SPC(11);TAB(17,20)SPC(2);
TAB(9,27);SPC(9);TAB(15,28);SPC(4)
 3090 ENDPROC
 3100 :
 3110 DEF PROCstop
 3120 COLOUR 3:COLOUR 128:K%=1
 3130 REPEAT:SOUND2,-15,RND(20)*4+53,2
 3140 A%=RND(10)
 3150 PRINTTAB(6,5);A%;" "
 3160 S=INKEY(20)
 3170 UNTIL S=70
 3180 PROCnudge(A%)
 3190 ENDPROC
 3200 :
 3210 DEF PROCnudge(A%):*FX15 1
 3220 ROLD1$=R$(1)
 3230 ROLD2$=R$(2)
 3240 ROLD3$=R$(3)
 3250 FOR B%=0 TO A%-1
 3260 PRINTTAB(6,5);A%-B%;" "
 3270 REPEAT:S=GET
 3280 UNTIL (S>48 AND S<52) OR (S>32 AN
D S<36)
 3290 T=INSTR("321 !""#",CHR$(S))-4:U=A
BS(T)
 3300 PR%(U)=PR%(U)+SGN(T)
 3310 IF PR%(U)>20 THEN PR%(U)=1:GOTO33
30
 3320 IF PR%(U)<1 THEN PR%(U)=20
 3330 IF PR%(U)>18 THEN R$(U)=RIGHT$(RL
$(U),21-PR%(U))+LEFT$(RL$(U),ABS(18-PR%
(U)))
 3340 IF PR%(U)<19 THEN R$(U)=MID$(RL$(
U),PR%(U),3)
 3350 FOR D%=1 TO 3
 3360 PRINTTAB(5*U-1,D%*3+5);F$(INSTR(S
$,MID$(R$(U),D%,1))-1)
 3370 NEXT:IF NOT(ROLD1$=R$(1) AND ROLD
2$=R$(2) AND ROLD3$=R$(3)) THEN PROCche
ck
 3380 IF GM%=1 B%=A%*2:F%=1
 3390 NEXT:PRINTTAB(6,5);SPC(3)
 3400 ENDPROC
 3410 :
 3420 DEF PROCchars:RESTORE 3490
 3430 FOR A%=224 TO 255:READ A$:VDU 23,
A%
 3440 FOR B%=1 TO 15 STEP 2
 3450 VDU EVAL("&"+MID$(A$,B%,2))
 3460 NEXT,
 3470 ENDPROC
 3480 :
 3490 DATA 0001030707070F0F00
 3500 DATA 0080C0E0E0F0F000
 3510 DATA 1F1F0F070300000
 3520 DATA F8F8F0E080804038
 3530 DATA 00070A1222524A47
 3540 DATA 00C0F0783C5C9E1E
 3550 DATA 7F474A5222120A07
 3560 DATA FF1E9E5C3C78F0C0
 3570 DATA 001D02C3E7FFFFF
 3580 DATA 000000000000080C0
 3590 DATA FFFFFFFF7F7F3F0F
 3600 DATA E0F0F0F8FCFEFEF8
 3610 DATA 000001F1F1F3FFF
 3620 DATA 78FCFCFEFFFFFFFF
 3630 DATA 3F1F1F0F01000000
 3640 DATA FFFFFFFEFCFC7800
 3650 DATA 558291A8908040A0
 3660 DATA 5582112811020508
 3670 DATA 55AA552A15081069
 3680 DATA 718041A0512845A3
 3690 DATA 061F3F766666663F
 3700 DATA C0F0F8DCCCC0C0F8
 3710 DATA 3F060666763F1F06
 3720 DATA F8CCCCCCDCF8F0C0
 3730 DATA 00FF80B690E61436
 3740 DATA 00FF01DD05D985DD
 3750 DATA 000A0A0A0A060000
 3760 DATA 016181BF90600000
 3770 DATA 0101010107081038
 3780 DATA 0000000080402010
 3790 DATA 387C7C3838100000
 3800 DATA 1038387C7C383810
 3810 DATA 2,3,5,2,7,2,5,1
 3820 :
 3830 ON ERROR OFF:MODE 7
 3840 IF ERR=17 END
 3850 REPORT:PRINT" at line ";ERL:END
```



IF YOU WRITE TO US

BACK ISSUES (Members only)

All back issues are kept in print (from April 1982) priced as follows:

Individual copies:
Volume 1 - £0.80
Volume 2 - £0.90
Volume 3 - £1.00

Volume 1 set (10 issues) £7
Volume 2 set (10 issues) £8

Please add cost of post and packing as shown:

No of copies	DESTINATION UK	Europe	Elsewhere
1	0.30	0.70	1.50
2 - 5	0.50	1.50	4.70
6 - 10	1.00	3.00	5.50
11 - 20	1.50	4.00	7.00

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.

This offer is 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

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

MEMBERSHIP COSTS:
U.K.
£6.40 for 6 months (5 issues)
£11.90 for 1 year (10 issues)

Eire and Europe
Membership £18 for 1 year.
Middle East £21
Americas and Africa £23
Elsewhere £25
Payment in Sterling essential.

PROGRAMS AND ARTICLES

All programs and articles used are paid for at around £25 per page, but please give us warning of anything substantial that you intend to write. In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "View", "Minitext Editor" or other means. 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.

Editorial Address

BEEBUG
PO Box 50
St Albans
Herts

Subscriptions & 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

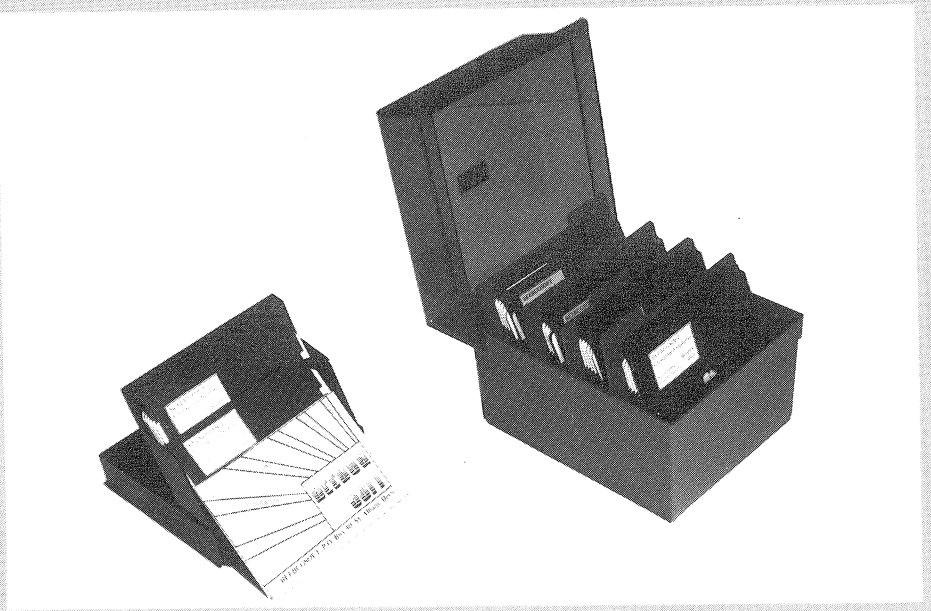
If you require members' discount on software it is essential to quote your membership number and claim the discount when ordering.

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Assistant Editor: Geoff Bains. Production Editor: Phyllida Vanstone.
Technical Assistants: David Fell and Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to Sheridan Williams, Adrian Calcraft, Matthew Rapier, John Yale, and Tim Powys-Lybbe for assistance with this issue.

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) 1984.

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

THE BEEBUG MAGAZINE ON DISC AND CASSETTE

The programs featured each month in the BEEBUG magazine are now available to members on disc and cassette.

Each month we will produce a disc and cassette containing all of the programs included in that month's issue of BEEBUG. Both the disc and the cassette will display a full menu allowing the selection of individual programs and the disc will incorporate a special program allowing it to be read by both 40 and 80 track disc drives. Details of the programs included in this month's magazine cassette and disc are given below.

Magazine cassettes are priced at £3.00 and discs at £4.75.
SEE BELOW FOR FULL ORDERING INFORMATION.

This Month's Programs Include:

Four Christmas Carols, a Cartoon Calendar for 1985, a colourful Fruit Machine game, an Improved Trace Facility for Basic programmers, three one line games – Asterisk Tracker, Truffle Hunt and Treasure Hunt, update to the Printer Spooler for Wordwise, two Workshop routines – Memory Display and Character Inverter, program for our Graphics Tablet construction project, demonstration of PLOT extensions, and as an extra, a superb 'Pengo'-style game with a Christmas flavour called Santa.

MAGAZINE DISC/CASSETTE SUBSCRIPTION

Subscription to the magazine cassette and disc is also available to members and offers the added advantage of regularly receiving the programs at the same time as the magazine, but under separate cover.

Subcription is offered either for a period of 6 months (5 issues) or 1 year (10 issues) and may be backdated if required. (The first magazine cassette available is Vol. 1 No. 10; the first disc available is Vol. 3 No. 1.)

MAGAZINE CASSETTE SUBSCRIPTION RATES

6 MONTHS	(5 issues) UK £17.00 INC... Overseas £20.00 (No VAT payable)
1 YEAR	(10 issues) UK £33.00 INC... Overseas £39.00 (No VAT payable)

MAGAZINE DISC SUBSCRIPTION RATES

6 MONTHS	(5 discs) UK £25.50 INC... Overseas £30.00 (No VAT payable)
1 YEAR	(10 discs) UK £50.00 INC... Overseas £56.00 (No VAT payable)

CASSETTE TO DISC SUBSCRIPTION TRANSFER

If you are currently subscribing to the BEEBUG magazine cassette and would prefer to receive the remainder of your subscription on disc, it is possible to transfer the subscription. Because of the difference between the cassette and disc prices, there will be an extra £1.70 to pay for each remaining issue of the subscription. Please calculate the amount due and enclose with your order.

ORDERING INFORMATION

Please send your order to the address below and include a sterling cheque. Postage is included in subscription rates but please add 50p for the first item and 30p for each subsequent item when ordering individual discs or cassettes in the UK. Overseas orders please send the same amount to include the extra post but not VAT.
SEND TO:

BEEBUGSOFT, PO BOX 109, HIGH WYCOMBE, BUCKS, HP10 8HQ

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

ISSN 0263-7561