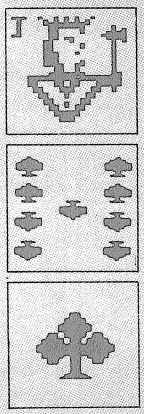
£1.00

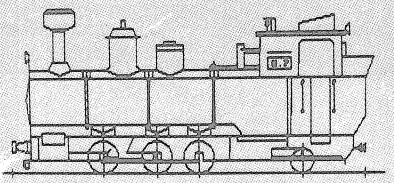
BEEBUG

FOR THE BBC MICRO

Vol 2 No 7 December 1983



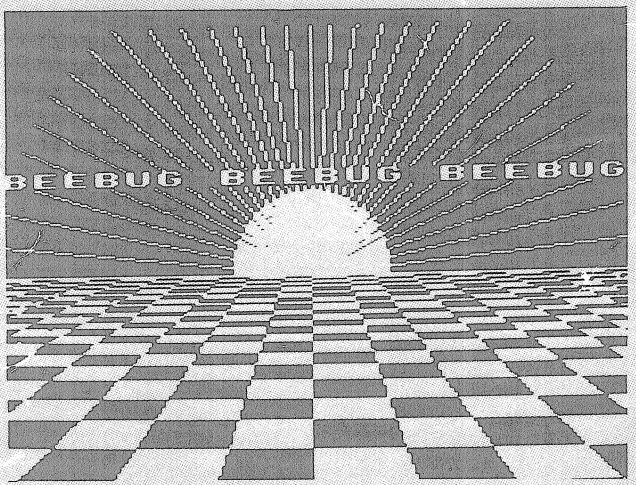
KILLER DICE



COMPUTER AIDED DESIGN

BOOK REVIEWS





CHEQUER BOARD

PLUS

- ***GALACTIC INVASION**
- ***LINK UTILITY**
- ***SPEAKER INTERFACE**
- ***PERCUSSION MACHINE**
- ***DISC DRIVES REVIEWED**
- ***And much more**

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 20,000

EDITORIAL

THIS MONTH'S MAGAZINE

In this month's issue, I would particularly draw your attention to our CAD pogram ASTAAD. As you will see from the illustrations, this program is capable of some remarkably sophisticated results for its short length. Because the program is controlled by function keys it will only run with O.S. 1.2 and this is also true, for the same reason, of our Killer Dice game. We believe that this is justified in the interests of presenting really good programs in the magazine. We said last month that we shall not be continuing support for O.S. Ø.1 after the end of 1983. In any case O.S. 1.2 ROMs are readily available from our software address. One further point on this month's programs is that both the games, Killer Dice and Galactic Invasion, use variable RESTORE statements, and will not work correctly if the DATA statements are renumbered.

SOFTWARE REVIEWS

This issue of the BEEBUG magazine is really packed with articles and programs, so much so that we had no space left for any software reviews. For this month, we have included some short notes on a number of software items in the supplement, and hope to return to fuller reviews in the main magazine next month.

MAGAZINE SUPPLEMENT

You may have noticed that from the last issue of BEEBUG, the supplement has BEEBUG/ELBUG printed at the bottom of each page. ELBUG is the name of the magazine that we are producing for members of ORBIT, our Electron User Group. The supplement will in future contain items relevant to both groups of users.

BEEBUG DIARY

You will notice elsewhere in this issue that we have extended the range of discounted items to members. We have also produced a special BEEBUG diary for 1984 which, amongst other things, contains all the information from the BEEBUG Reference Card that was originally published with Vol.2 No.1.

ACORN EDUCATION EXHIBITION

We shall have a stand at this January event (see Events in the supplement) and hope to see many of our educational readers there.

BEEBUG ON TV AGAIN

Finally, BEEBUG has made another TV appearance. In the BBC TV programme "Tomorrow's World" on the 3rd November the presenter, Judith Hahn, demonstrated the 3D Rotation program by James Hastings from BEEBUG Vol.1 No.1Ø. It was used as an example of 3D computer graphics.

Mike Williams

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOAI

HINT WINNERS

This month's hint winners are D.E.Susans who wins the £1Ø prize, and J.Pike and N.Ahmon who both win a £5 prize. Keep sending those hints in please.

MAGAZINE CASSETTE

This month's magazine cassette contains just one extra item, a demonstration version of Alien Destroyer, one of the new BEEBUGSOFT games.

BEEBUG MAGAZINE

GENERAL CONTENTS

PAGE	CONTENTS
2	Editorial
4	ASTAAD - A Computer Aided Design Program
8	The Twelve Days of Christmas
10	The Teletext Mode (Part 3)
14	Points Arising
15	Five Books for Christmas
17	Screen Freezer
19	Fitting an External Speaker Interface to the Beeb
22	The Percussion Machine
24	A Link Utility for Programmers
28	Moving Chequer Board
30	Three Disc Drives Reviewed
33	Killer Dice
37	Postbag
38	Galactic Invasion

HINTS, TIPS & INFO

PAGE	CONTENTS
7	Tape to Disc Transfer
16	INPUTLINE
18	Useful Key Definition for WORDWISE
18	Upper/Lower Case Characters
21	Bigger Basic Programs in Modes 0, 1 & 2
21	Summary of Start-up Options
21	Null Filenames
21	Two Useful Locations in O.S. 1.2
23	CLG Bug
32	*FX3,10
32	Auto Version Numbering
36	Colouring VIEW
36	COUNT Errors
36	Detecting the Shift and Control Keys
36	O.S. 1.2 Keyboard Scan
36	Speeding up A/D Conversion

PROGRAMS

PAGE	DESCRIPTION
4	Computer Aided Design
8	Twelve Days of Christmas
10	Teletext Flags
10	Teletext Flash
10	Teletext Fish1
10	Teletext Fish2
17	Screen Freezer
22	Percussion Machine
24	Link Utility
28	Chequer Board
33	Killer Dice
38	Galactic Invasion

ASTAAD – A COMPUTER AIDED DESIGN PROGRAM (32k)

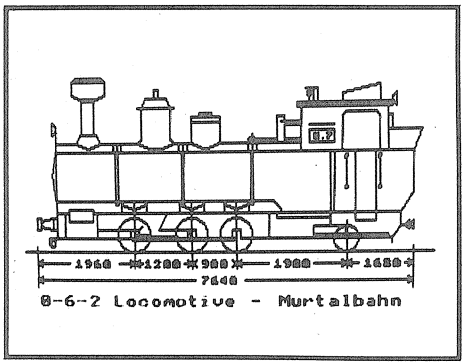
Basic I & II
O.S. 1.2 only

by Tim Tonge

Computer Aided Design (CAD) is one of the more fascinating applications of computers and one for which the BBC micro has considerable potential. We present here a really excellent example of a CAD package which contains all the basic drawing facilities. Not only that, but the package will also allow you to produce Any Size Text printed at any Angle Anywhere on the drawing, which explains the origins of the name of the package, ASTAAD.

INTRODUCTION

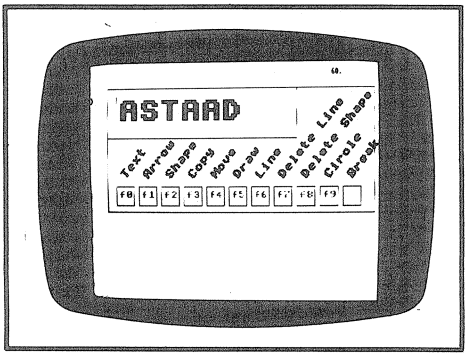
As you can see from the illustrations accompanying this article, the ASTAAD program is capable of producing remarkably sophisticated drawings for its extremely compact size. The program is also well structured so that you can readily extend the range of facilities to suit your own needs. The program runs in mode 0 for the maximum resolution, and all the drawing is done in black on a white background. The various options are selected by the function keys plus Tab and Copy.



0-6-2 Locomotive - Murtalbahn

USING THE PROGRAM

When the program is RUN, a flickering cursor appears at the centre of the screen, which can be moved in steps of 10 g.u.(graphics units) by the four cursor keys. Function key f5 makes the cursor DRAW, f7 makes it DELETE, and f4 restores MOVE. The distance from the starting point is shown in the top right hand corner, and this acts as a ruler. This can be reset to a new starting point at any time by pressing Tab, and this also happens automatically when you change direction using the cursor keys. The ruler is very useful for measuring parts of a drawing and positioning new objects.



Another method of drawing a line, at an any angle, is to move to one end of the line and press Tab, and then move the cursor to the other end, and press Copy. To delete the line, press Copy again. For greater accuracy, after moving to one end of the line, press f6 and then enter the length in g.u. and angle in degrees, each followed by Return. For example, f6 234 <return> 90 <return> gives a vertical line from the cursor towards the top of the screen; f6 432 <return> 180 <return> gives a horizontal line to the left. Note that Return on its own enters zero, so that f6 432 <return> <return> draws a horizontal line to the right. Also both positive and negative angles can be specified.

The standard characters can be entered directly from the keyboard, the cursor being moved after each one, normally two steps to the right. Underlining needs one step down, superscripts one step up, and the characters ^ / and \ can be positioned for accents.

"ASTAAD" itself is brought into use by f0 after moving the cursor to the required position, and 3 inputs are called for. The text input is limited

to 41 characters. In response to "Size" any number can be used, but between 2 and 3 is the smallest likely to be legible, depending on the TV/monitor in use, while 125 almost fills the screen with one character. Any angle can be entered, positive or negative. Note that text can be underlined using the Tab-Copy feature or the f6 routine. Many interesting effects can be discovered by experiment. For example, a row of "size 5 underlines" gives a useful thick line, and with size 8 you can draw a frame around your drawing.

Arrows are often needed on drawings for dimensioning and on leader lines pointing to objects. Function key f1 draws an arrow at the cursor position after the direction angle has been entered.

Function key f2 produces circles, ellipses, triangles, rectangles, pentagons and other polygons to be drawn around the cursor. Three inputs are called for, Horizontal Axis (in g.u.), Vertical Axis, and Number of Sides. A few examples show the way:-

```
Circle, radius 100g.u.:
100 <return> 100 <return> 30 <return>
Ellipse:
200 <return> 50 <return> 30 <return>
Square, length of side 300g.u.:
300 <return> 300 <return> 4 <return>
Regular pentagon:
500 <return> 500 <return> 5 <return>
Vertical line:
<return> 82 <return> 2 <return>
```

Function key f3 enables you to repeat or copy the last shape (produced using f2) at a new cursor position, and is useful for windows in houses, portholes in ships, shading with dots and angled lines, and so on. If a mistake is made f7 followed by f3 will delete the shape just drawn.

Circles are needed so frequently that f9 is programmed so that only the radius need be entered. Radius=0 (i.e.f9 <return>) prints a dot, and f9 5 <return> produces a useful blob.

To clear an area of the screen, move the cursor to the bottom left hand corner and press TAB, and then move to the top right hand corner and press f8. This will clear the rectangular area defined by these two opposite corners. Reversing the two corners will usually clear the whole screen so be careful! If you want to clear the whole screen then Ctrl-L is the quickest way. Escape can be used if a mistake is made in the middle of a procedure, or to move the cursor to the centre of the drawing area.

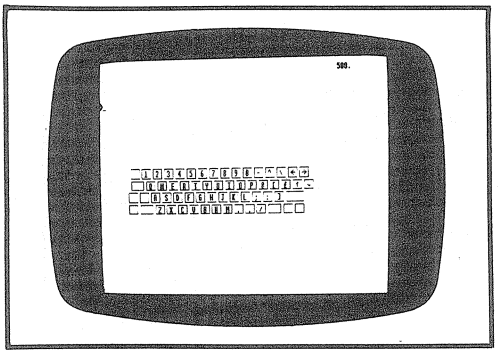
PROGRAM NOTES

Mode 0 is used in the program for maximum resolution and PROCsetup, called at the start of the program initialises the variables used and positions the cursor at the centre of the screen. *FX4,2 makes the 4 cursor keys and Copy generate ASCII codes, and *FX225,129 causes the function keys to produce ASCII codes 129 upwards. X% and Y% are the cursor co-ordinates with initial mid-screen values.

The main program repeatedly looks for keyboard input while at the same time maintaining the cursor on the screen. Whenever a key is pressed the program distinguishes between normal ASCII codes

f9 CIRCLE | f8 DELETE AREA | f7 DELETE LINE | f6 LINE | f5 DRAW | f4 MOVE | f3 REPEAT f2/f9 | f2 SHAPE | f1 ARROW | f0 ASTAAD

(less than 128 in value) which are sent direct to the screen (these are mainly printing characters plus the usual control characters), and other inputs such as the function keys, cursor keys, Tab and Copy. For these the program branches to the corresponding procedures and line drawing routines depending on the value of J% (in the range 1 to 16). All cursor keys movements are handled by routines starting at line 250.





```
  10 REM Program ASTAAD
  20 REM Version B1.2
  30 REM Author Jim Tonge
  40 REM BEEBUG December 1983
  50 REM Program subject to copyright
  60 :
  70 MODE 0:ON ERROR GOTO 1010
  80 PROCsetup
  90 REPEAT
 100 PROCruler
 110 key=INKEY(0):IFkey<=0 THEN 270
 120 IF key=9 THEN V=5:X1%=X%:Y1%=Y%:k
=10:Z=4
 130 IF key<129 PRINT CHR$(key):X%=X%+
20*(X%<1250)*(key<>9):GOTO110
 140 J%=key-128
 150 ON J% GOTO 180,160,200,220,230,23
0,170,230,190,210,110,260,250,250,250,2
50 ELSE 150
 160 PROCarrow:PROCcursor
 170 PROCline:PROCcursor
 180 PROCtext:PROCcursor
 190 PROCdelete:PROCcursor
 200 PROCpoly:PROCshape(H%,V%,G%):PROC
cursor
 210 PROCcircle:PROCshape(H%,V%,G%):PR
OCcursor
 220 PROCshape(H%,V%,G%):PROCcursor
 230 IF J%>4 AND J%<7 OR J%=8 THEN Z=J
%-1 ELSE Z=4
 240 IF J%=8 Y=7 ELSE Y=5:y=Y-4
 250 X%=X%-10*(J%=13)*(X%>10)+10*(J%=1
4)*(X%<1260):Y%=Y%-10*(J%=15)*(Y%>10)+1
0*(J%=16)*(Y%<990):PLOTZ,X%,Y%
 260 IF J%=12 k=0:PLOTV,X1%,Y1%:IF V=5
 THEN V=7 ELSE V=5
 270 PROCcursor
 280 UNTIL FALSE
 290 END
 300 :
 310 DEFPROCsetup
 320 VDU5
 330 VDU23;8202;0;0;0;
 340 VDU19,0,7;0;19,7,0;0;
 350 *FX4,2
 360 *FX225,129
 370 X%=630:Y%=490:Y=5:Z=4:J%=0:k=0:B%
=0:T%=0:j=0:X1%=630:Y1%=490
 380 ENDPROC
 390 DEFPROCruler
 400 IF J%<13 OR J%>18 ENDPROC
 410 IF k=10 THEN 430 ELSE 420
 420 IF J%<>j THEN X1%=X%:Y1%=Y%
 430 B%=SQR((X1%-X%)^2+(Y1%-Y%)^2)
 440 VDU4:@%=131082:PRINTTAB(60,1)B%+1
0-k:VDU5:j=J%
 450 ENDPROC
 460 DEFPROCtext
 470 IF J%<>1 ENDPROC
 480 VDU4,28,6,1,59,1:INPUTTAB(6,1)"Te
xt? "T$,"Size? (2 to 125):  "S%,"Angle(
deg.)?"T%:CLS:VDU5,26
 490 E=S%*SIN(RAD(T%)):F=S%*COS(RAD(T%))
 500 FOR C%=1 TO LEN(T$)
 510 A%=&BF00 +ASC(MID$(T$,C%,1))*8
 520 FOR P% =0TO 7:FOR Q% =0 TO 7
 530 MOVE X%+P%*E-Q%*F+C%*7.6*F,Y%-Q%*
E-P%*F+C%*7.6*E
 540 IF S%<5 THEN PROCdot ELSE PROCsq
 550 NEXT Q%,P%,C%
 560 ENDPROC
 570 DEFPROCdot
 580 IF (2^Q% AND A%?P%)<>0 THEN PLOT6
5,0,0
 590 ENDPROC
 600 DEFPROCsq
```

```
  610 IF (2^Q% AND A%?P%)<>0 THEN PLOT0
,F,E:PLOT81,(E-F),-(E+F):PLOT81,F,E
  620 ENDPROC
  630 DEFPROCarrow
  640 IF J%<>2 THEN ENDPROC
  650 VDU4,28,6,1,59,1:INPUTTAB(6,0)"An
gle of arrow?  "W%:CLS:VDU5
  660 a=15*COS(RAD(W%+35)):b=15*SIN(RAD
 (W%+35)):c=15*COS(RAD(W%-35)):d=15*SIN
(RAD(W%-35))
  670 PLOTy,-a,-b:PLOT0,a,b:PLOTy,-c,-d
:PLOT0,c,d:VDU26
  680 ENDPROC
  690 DEFPROCline
  700 IF J%<>7 THEN ENDPROC
  710 VDU4,28,6,1,59,1:INPUTTAB(6,0)"Le
ngth? "L%,"Angle?(deg.) "N%:CLS:VDU5
  720 LX=L%*COS(RAD(N%)):LY=L%*SIN(RAD(
N%)):X2%=X%+LX:Y2%=Y%+LY:PLOTY,X2%,Y2%:
X%=X2%:Y%=Y2%:VDU26
  730 ENDPROC
  740 DEFPROCpoly
  750 IF J%<>3 THEN ENDPROC
  760 VDU4,28,6,1,59,1:INPUTTAB(6,0)"Ho
rizontal Axis?"H%,"Vertical Axis?"V%,"N
o.of sides(30=circle or ellipse)?"G%:CL
S:VDU5,26
  770 IF G%=4 THEN H%=H%/SQR(2):V%=V%/S
QR(2)
  780 ENDPROC
  790 DEFPROCshape(H%,V%,G%)
  800 IF NOT(J%=3 OR J%=4 OR J%=10)THEN
 ENDPROC
  810 LOCAL I%,K%,M%,N,C,S,B,D,R,T
  820 N=2*PI/G%:C=COS(N): S=SIN(N):B=1/
SQR(2):D=1/SQR(2)
  830 FOR I%=1 TO G%+1
  840 R=B*C-D*S:T=B*S+D*C
  850 B=R:D=T:K%=H%*B+X%:M%=V%*D+Y%
  860 IF I%>1 THEN PLOTY,K%,M% ELSE MOV
E K%,M%
  870 NEXTI%
  880 ENDPROC
  890 DEFPROCcircle
  900 IF J%<>10 THEN ENDPROC
  910 VDU4,28,6,1,59,1:INPUTTAB(6,0)"Ra
dius of circle? "R%:CLS:VDU5,26
  920 H%=R%:V%=R%:G%=30
  930 ENDPROC
  940 DEFPROCdelete
  950 IF J%<>9 ENDPROC
  960 VDU24,X1%;Y1%;X%;Y%;:CLG:VDU26
  970 ENDPROC
  980 DEFPROCcursor
  990 PLOT4,X%,Y%+15:PLOT6,X%,Y%-15:PLO
T 4,X%-15,Y%:PLOT6,X%+15,Y%:PLOT4,X%,Y%
 +15:PLOT6,X%,Y%-15:PLOT4,X%-15,Y%:PLOT
6 ,X%+15,Y%:PLOT4,X%,Y%
 1000 ENDPROC
 1010 IF ERR<>17 THEN ON ERROR OFF:MODE
7:REPORT:PRINT" at line ";ERL:END
 1020 GOTO 80
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TAPE TO DISC TRANSFER - J.A.Knaggs

It is possible to have a cassette file and a disc file open at the same time. Using this facility it is possible to transfer files from tape to disc without having to load the file into memory. It is necessary to execute *TAPE and *DISC to select filing systems as required e.g.

```
10 *TAPE
20 IN=OPENIN("ADVENTR")
30 *DISC
40 OUT=OPENOUT("ADVENTR")
50 *TAPE
60 REPEAT
70 C=BGET#IN
80 *DISC
90 BPUT#OUT,C
100 *TAPE
110 UNTIL EOF#IN
120 CLOSE#IN
130 *DISC
140 CLOSE#OUT
```

Having performed this operation it is then necessary to set the load and execute addresses of the file. This can be done with OSFILE calls 2 and 3. To set a load address of &1900 and an execute address of &1D82 use:

```
10 OSFILE=&FFDD
20 DIM PARAM 18,NAME 8
30 $NAME="ADVENTR"
40 !PARAM=NAME
50 PARAM!2=&1900
60 PARAM!6=&1D82
70 X%=PARAM MOD 256
80 Y%=PARAM DIV 256
90 A%=2
100 CALL OSFILE
110 A%=3
120 CALL OSFILE
```

Tested on O.S. 0.1 and 1.2 and on Basics I and II

THE TWELVE DAYS OF CHRISTMAS (16k)

by Douglas Nunn







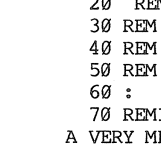


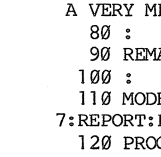


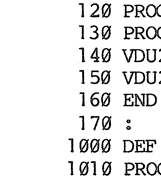
Here is a program with a really seasonal flavour. Often we are unable to publish good music programs in the magazine because of their length or because of the problems of copying multitudes of data staements with accuracy. This program is both short and easy to copy. The program will provide you with an excellent rendering of "The 12 Days Of Christmas" complete with verses that scroll up the screen in time with the music. If you provide your micro with a loudspeaker interface, as described elsewhere in this issue, then you could be really surprised at the musical quality of the BBC micro. Even without an external speaker, the results are impressive.

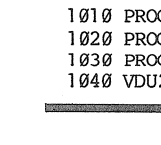
The program itself is quite interesting. The music is generated in three part harmony using a coding system. This is the function of the procedure PROCplay, which appears frequently in the program. The three parameters of the procedure code the notes played by the three music channels. When you are typing in the program it is important to do this accurately, especially where the parameter strings for PROCplay are concerned, if you are to obtain the best results. Spaces are significant here. The program also very neatly links the playing of the notes with the display of the verses on the screen.

Once the program is loaded it is quite interesting to experiment with it in immediate mode. Just type in 'PROCstart' to set everything up and then calling other procedures like PROCfive or PROCtwelve will produce appropriate parts of the music. The variable D% specifies the verse number (1 to 12) and Z% controls the speed of the music (4 is the normal speed, increasing the value decreases the speed). Varying the value of T% (set to 139 initially) will alter the octave range.

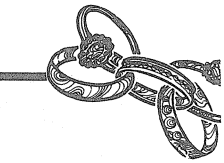






















```
   10 REM Program 12 DAYS OF CHRISTMAS
   20  REM Version B0.4
   30 REM Author Douglas Nunn
   40 REM BEEBUG December 1983
   50 REM Program subject to Copyright
   60 :
   70 REMISS OF ME NOT TO WISH EVERYONE
 A VERY MERRY XMAS AND A HAPPY NEW YEAR
   80 :
   90 REMARKABLY WELL PUT - Ed.
  100 :
  110 MODE 7:ON ERROR ON ERROR OFF:MODE
7:REPORT:PRINT" at line ";ERL:END
  120 PROCstart
  130 PROCmusic
  140 VDU28,0,24,39,0
  150 VDU23;11,255,0;0;0;
  160 END
  170 :
 1000 DEF PROCstart
 1010 PROCinit
 1020 PROCdata
 1030 PROCfrontpage
 1040 VDU23;11,0,0;0;0;
 1050 ENDPROC
 1060 :
 1070 DEF PROCinit
 1080 DIM A$(10),S$(2),P$(11)
 1090 Z%=4:WIDTH0
 1100 FOR E%=6TO8:REPEAT UNTIL ADVAL(-E
%)=15:NEXT
 1110 ENDPROC
 1120 :
 1130 DEF PROCdata
 1140 FOR E%=0TO10:READS$:A$(E%)=S$+","
+CHR$10+CHR$13:NEXT
 1150 A$(3)="Five"+CHR$131+"gold"+CHR$1
35+"rings,"+CHR$10+CHR$13
 1160 FOR E%=0TO11:READ P$(E%):NEXT
 1170 T%=139
 1180 ENDPROC
 1190 :
 1200 DEF PROCfrontpage
 1210 FORI%=0TO1:PRINTTAB(0,I%);:VDU129
,157,130,141:PRINTTAB(8,I%);"Twelve Day
s of Christmas";:NEXT I%:VDU28,2,24,39,
2
```

»

```
 1220 REPEAT:CLS:INPUT'"Enter volume (
1=soft : 15=loud )"V%:UNTIL V%<16 AND V
%>0:CLS
 1230 ENDPROC
 1240 :
 1250 DEF PROCmusic
 1260 FOR D%=1TO12
 1270 IF D%=12THEN PROCtwelve:GOTO1310
 1280 PROCfirst:IF D%=1THEN1330
 1290 IF D%<5 PROCplay("@-","7-","$-"):
FOR E%=D%TO2STEP-1:PROCunder5:PRINTA$(E
%-2);:NEXT:PROCplay("@-",";-","7-"):GOT
O1330
 1300 PROCplay("@-","7-","$-")
 1310 IF D%>5 THEN FOR E%=D%TO6STEP-1:P
ROCover5:NEXT
 1320 PROCfive
 1330 IF D%=12 PROCplay("@-C-H-H-H!","4
-<-@-@-@!","+-(-$-$-$!"):PRINT'TAB(14)"
< THE END "; ELSE PROClast:IF (D%+1)MOD
3=0 PROCplay("<-<-","6-6-","-,-,-"):T%=T
%-4 ELSE PROCplay("<-<-","4-4-","$-$-")
 1340 NEXT D%
 1350 ENDPROC
 1360 :
 1370 DEF PROCtwelve
 1380 Z%=8:PROCplay(" -7!7!7-7-<!<!","
-7!5!4-5-4-7-"," - - - -2-0-/-"):PRINT'
'"On the"CHR$129P$(D%-1)CHR$135"day of
Christmas"'" my true love gave to me,"
 1390 PROCplay("<-<!<-||<!","<-;-9-||7-
","--+-)-||(-")
 1400 Z%=Z%/2:PROCplay(">-><-@-@-A-A->->
-","9-;-9-7-8-:-8-6-","-)-+-)-(-)-+-,-.-
"):Z%=Z%*3:PROCplay("@-@-@-@-@-@-@-@-@-",
"7-;-9-<-6-6-6-6-","0!0-0---&-&-&-&-")
 1410 PROCplay(" -","  -","  -"):Z%=2
 1420 ENDPROC
 1430 :
 1440 DEFPROCfirst
 1450 PROCplay("7!7!7-7-","  - -4-4-","
- -$-$-")
 1460 PRINT'"On the"CHR$(129+D%MOD6)P$
(D%-1)CHR$135"day of Christmas"'" my tr
ue love gave to me,"
 1470 PROCplay("<!<!<-<!<!<!>-@-A->-@-@
-@-","  - -4-4- -;-9-9- -5-7-7-7-","  - -
$-$-  - -$-$-  -+-$-$-$-")
 1480 ENDPROC
 1490 :
 1500 DEFPROClast
 1510 PROCplay("A-C-C-E-",";-<-;-9-","&
-(-(-)-")
 1520 IFD%-1PRINT"and ";
 1530 PRINT"a partridge in a pear tree.
"
 1540 PROCplay("A-@-<->->-<-<-<-<-","><
-<-7-7-5-4-4-4-4-","-)-+-+-+-+-$-$-$-$-")
 1550 ENDPROC
 1560 :
 1570 DEFPROCunder5
 1580 PROCplay("C-C->-@-A-A-","<-<->-<-
;-;-","(-(-)-)-+-+-")
 1590 ENDPROC
 1600 :
 1610 DEFPROCfive
 1620 IFD%=12 THEN Z%=4
 1630 PROCplay("C-C-C-C-E-",";-;-;-;-9-
","(-(-(-(-&-"):PRINTA$(3);:PROCplay("B
-B-B-C-C-C-C-C-C-C-C-","6->-<-;-9-;-<->
-<-;-9-","&-&-&-+-&-+---/---+-)-")
 1640 IF D%=12 THEN Z%=6
 1650 PROCplay("C-A-@->-<-<-",";-;-7-;-
9-9-","(-(-(-(-----"):PRINTA$(2);:PROCp
lay("A-A-9-9-<-<-","  - -5-5-7-7-","  - -
&-&-(-(-"):PRINTA$(1);:IF D%=12 GOTO167
0 ELSE PROCplay(">-<-;-9-7-","6-4-2-0-/
-","*-*-&-&-+!"):PRINTA$(0);:PROCplay("
7-@-","2-7-","+-+-")
 1660 ENDPROC
 1670 PROCplay(">-<-;-9-7-","2-7-7-0-/-
","&-'-(-*-+!"):PRINTA$(0);:PROCplay("
-@-A-C-C-E-A-","  -7-9-:!:-9-<-","+-+-)-
(-(-)-&-")
 1680 PRINT"and a partridge in a pear t
ree."
 1690 PROCplay("@-<->->-<-<-<-<-<!","7-
9-;-5-4-4-4-4-4-4!","+-+!+-+-$-$-$-$-$!
")
 1700 ENDPROC
 1710 :
 1720 DEFPROCover5
 1730 PROCplay("C-C->-@-A-","<-9-9-:-;-
","(-)-)-*-+-"):PRINTA$(E%-2);:PROCplay
(">-","-;!","+-")
 1740 ENDPROC
 1750 :
 1760 DEFPROCplay(S$(0),S$(1),S$(2))
 1770 LOCAL E%,F%,N%,P%,Q%,N%,V
 1780 FOR E%=1TOLENS$(0)STEP2
 1790 FOR F%=0TO2
 1800 P%=ASC(MID$(S$(F%),E%)):Q%=ASC(MI
D$(S$(F%),E%+1))
 1810 N%=4*P%-T%:V=V%*(P%<>32)
 1820 IF Q%=ASC"|" SOUND1+F%,0,0,1:GOTO
1840
 1830 IF Q%=ASC"-" SOUND1+F%,V,N%,Z% EL
SE SOUND1+F%,V,N%,Z%-1:SOUND1+F%,0,0,1
 1840 NEXT F%:NEXT E%
 1850 ENDPROC
 1860 :
 1870 DATATwo turtle doves,Three french
 hens,Four calling birds,!,Six geese a-
laying,Seven swans a-swimming,Eight mai
ds a-milking,Nine ladies dancing,Ten lo
rds a-leaping,Eleven pipers piping,Twel
ve drummers drumming
 1880 DATAfirst,second,third,fourth,fif
th,sixth,seventh,eight,ninth,tenth,elev
enth,twelfth
```

Tested on O.S. 0·1 and 1·2
and on Basics I and II

THE TELETEXT MODE (PART 3)

by Mike Williams

In our previous two articles on the Teletext Mode we have introduced the basic features of both text and graphics. This month we cover some less well known features of the Mode 7 and look at some useful programming techniques.

Those of you who have read both of the two previous articles in this series should now be familiar with the use of both text and graphics in this particular mode. If you have referred to the User Guide about Teletext Mode then you may well have noticed that there are still a few control characters that we haven't yet described.

HOLD GRAPHICS CHARACTER

The first new control character is called 'hold graphics' and has the value of 158. You should by now be familiar with the fact that every Teletext control character occupies a position on the screen which normally appears as a space. Often this space will cause us no problems at all, particularly when dealing with text, where words are usually separated by spaces anyway, and thus control characters can easily be slipped in between, appearing on the screen as the spaces we require.

With graphics the problem of an unwanted space is more likely to arise. Try typing the following line, in immediate mode, into your micro:

```
PRINT CHR$145;STRING$(36,CHR$240)
```

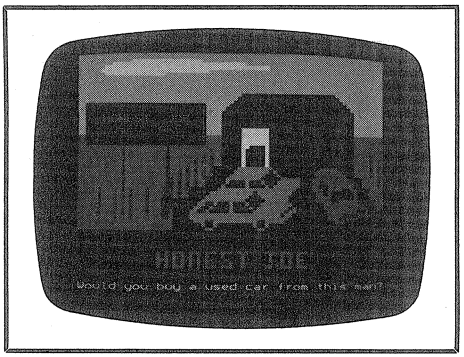
The graphics character with value 240 is a short horizontal line one pixel in height. You should find that the instruction produced a line of these characters across the screen in red (because of the 145 control code).

Now suppose that what we really want is a line which is half red and half blue. Try the following instruction:

```
PRINT CHR$145;STRING$(18,CHR$240);CHR$148;STRING$(18,CHR$240)
```

You should now find that the first half of the line is in red, and the second half of the line is in blue. Unfortunately, the two halves are separated by a space. This is where the 'hold graphics' code (158) can help. Try typing this variation of the last instruction into the computer:

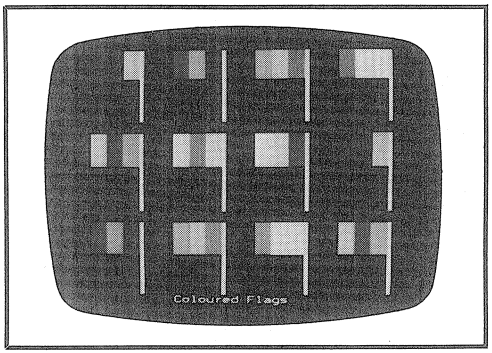
```
PRINT CHR$158;CHR$145;STRING$(17,CHR$240);CHR$148;STRING$(18,CHR$240)
```



You should now find that you have displayed on the screen exactly what we have been trying to achieve, a line of graphics characters which is half red and half blue with no space in between. Once the 158 control code has been placed on a line on the screen, any subsequent graphics control character is represented by the last displayed graphics character instead. This is why in the example above, the number of graphics characters in the first half of the line was reduced from 18 to 17 as the 148 control code is represented on the screen by a copy of the last graphics character, another 240. There is a second control character called 'release graphics' with a value of 159. This is used to cancel the effect of a 'hold graphics' character placed earlier on the same screen line.

This use of the 'hold graphics' character can be utilised wherever there is the need for adjacent graphics characters to be of different colours, and is illustrated in the program, called FLAGS, which displays on the screen a series of randomly coloured

and striped flags. The 'hold graphics' character is needed to allow the adjacent vertical stripes to be of different colours and different again from the flagpole. If you type in the program and run it you will see the colourful results produced.




```
   10 REM Program FLAGS
   20 REM Version B1.1
   30 REM Author Mike Williams
   40 REM BEEBUG December 1983
   50 :
  100 MODE7:DIM col(3)
  110 ON ERROR GOTO 240
  120 FOR I=0 TO 24:VDU158,13,10:NEXT I
  130 :
  140 x=2:y=1:c1=RND(7)-1:c2=RND(7)-1:c
3=RND(7)-1
  150 REPEAT
  160 PROCflag(x,y,c1,c2,c3)
  170 c1=(c1+1)MOD7:c2=(c2+2)MOD7:c3=(c
3+3)MOD7:x=x+10
  180 IF x>35 THEN x=2:y=y+8
  190 UNTIL y>20
  200 PRINT TAB(12,24);"Coloured Flags";
  210 REPEAT UNTIL FALSE
  220 END
  230 :
  240 ON ERROR OFF:MODE7
  250 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
  260 END
  270 :
 1000 DEF PROCflag(x,y,col(1),col(2),co
l(3))
 1010 LOCAL c,i
 1020 FOR i=y TO y+2
 1030 PRINT TAB(x-1,i);
 1040 FOR c=1 TO 3
 1050 PRINT CHR$(col(c)+145);CHR$(255);
 1060 NEXT c,i
 1070 FOR i=y TO y+6
 1080 PRINT TAB(x+5,i);CHR$151;CHR$181;
 1090 NEXT i
 1100 ENDPROC
```

PROGRAM NOTES

Each flag consists of three vertical stripes with the colours chosen somewhat randomly. Examples will normally appear in which two of the colours are the same or the same as the flagpole itself, which is always in white. Each flag is drawn by a procedure called PROCflag, defined in lines 1000 to 1100. The procedure uses 5 parameters, the position of the top left hand corner of the flag (specified as x and y for horizontal and vertical positions), and the colours for the three stripes. The colours are specified as a number in the range 0 to 6, and then 145 is added to this within the procedure.

Each flag is 6 characters long by 3 characters high, plus the flagpole which is 7 characters high. Each row of the flag is printed in turn and consists of a colour control character followed by the graphics character 255 (all 6 pixels) for each of the three colours in turn. The flagpole is produced separately and consists of a double column of 'white' graphics (151) followed by the graphics character 181, which is one pixel wide and three pixels high. With nothing else in the program the stripes in the flag would be separated by spaces marking the positions of the control characters. The main program, however, starts off by placing a 'hold graphics' character at the start of each line on the screen, in line 120, so that each colour control character is represented on the screen by a repeat of the preceding graphics character.

Line 140 selects the starting position for the first flag and selects three colours at random (the values c1, c2 and c3). The loop from line 150 to 190 then displays each flag in turn, changing position and changing the colours of each flag. You can terminate the program by pressing Escape.

CONCEAL DISPLAY CHARACTER

The second new control character this month is called the 'conceal display' character and has a value 152. Once placed on the screen everything

else following on that line will be hidden from view until the next colour control character. This allows items on the screen to be concealed, and revealed only when the 152 control character is overwritten by another character. This can be very useful in games like battleships where you are trying to locate hidden objects. The same feature can also be used to make things flash on the screen at a rate controlled by the program. Try typing the following short program FLASH into your micro, and then running it:

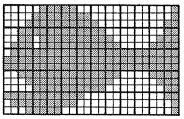
```
10 MODE7
20 PRINT TAB(5,5);"Flashing"
30 FOR I=1 TO 500
40 PRINT TAB(4,5);
50 IF I MOD 2=0 THEN VDU152 ELSE VDU
133
60 Z=INKEY(50)
70 NEXT I
80 END
```

The word 'Flashing' is displayed on the screen. By alternately preceding this with either the 'conceal display' character (152) or magenta text (133), the word appears to flash. The rate of flashing is controlled by the value in the INKEY statement at line 60. Larger values will produce a slower rate of flashing and smaller values a higher rate. Try experimenting with this and see the results.This can be achieved more easily using code 136 for flashing text, although you cannot now control the rate of flashing.

SOME INTERESTING GRAPHICS TECHNIQUES

In the remainder of this month's article I want to look at some of the problems that can arise in using Teletext Mode graphics, and some of the techniques that can help. Those of you with access to Ceefax, Oracle or Prestel (which all use Teletext Mode displays) will have seen some of the graphics designs that can be produced, some good and some bad! Designing good Teletext graphics displays is not an easy job.

By way of illustration I shall take as my task the design of a fish, which I want to display anywhere I choose on the screen and in any colour. This might be the basis of some simple computer game or as part of the front page display of a program. Using BBC Basic it is sensible to define a procedure, with parameters for the colour and the position of the fish, which should then make it easy to place the fish anywhere on the screen and in any colour. Before I can do that I also need to define the fish in some way and this will be best achieved with another procedure. In the absence of a good Teletext Editor, the only way I know is to sit down with squared paper and a pencil and draw roughly the shape you want. It is sensible to decide on a grid of characters that will contain the shape and I chose 5 rows each of 12 characters. You can see my design in the diagram.



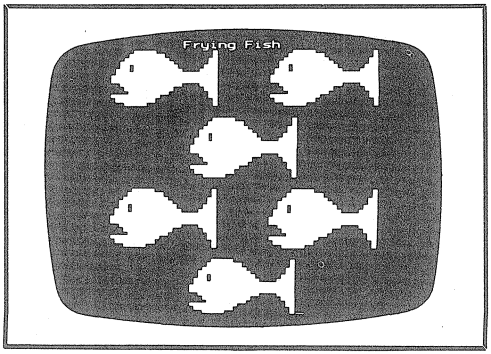
The next step was to work out, with the help of the User Guide, all the corresponding graphics codes. These are contained in 5 lines of DATA statements in the program. The procedure to define the fish establishes an array called fish$(5) which is then programmed so that each of the array elements from fish$(1) to fish$(5) contains a string of 12 characters for rows 1 to 5 of the fish. I can now write the procedure to display the fish. This will print each row of the fish in the right position on the screen, each row being preceded by the appropriate graphics colour control code. Here, one small problem arises. If the procedure specifies (x,y) as the position of the fish, should this be literally where the fish itself starts, or should this first position contain the colour control code, which won't of course be visible on the screen. There is no single answer to this, but whichever choice you make, you must follow this consistently. I prefer to place a shape in the (x,y) position and hence the control code goes into position (x-1,y). If you do this you must now make sure that you don't position a shape too close to the left hand side of the screen, so that position (x-1,y) is off the screen. Here then are the

»

two procedures to define and place a fish, complete with the DATA statements.

```
1000 DEF PROCdefinefish
1010 LOCAL i,j,f%,fish$
1020 FOR i=1 TO 5:fish$=""
1030 FOR j=1 TO 12
1040 READ f%:fish$=fish$+CHR$(f%)
1050 NEXT j:fish$(i)=fish$
1060 NEXT i
1070 ENDPROC
1080 :
1090 DEF PROCplacefish(c,x,y)
1100 LOCAL i,j
1110 FOR i=1 TO 5
1120 PRINT TAB(x-1,i+y-1);CHR$(144+c);
fish$(i);
1130 NEXT i
1140 ENDPROC
1150 :
1160 DATA 160,224,254,255,255,252,176,
160,160,160,160,250
1170 DATA 224,255,235,255,255,255,255,
244,160,160,248,255
1180 DATA 254,255,255,255,255,255,255,
255,255,255,255,255
1190 DATA 160,171,255,255,255,255,191,
163,160,160,171,255
1200 DATA 171,239,255,255,175,161,160,
160,160,160,160,235
```

I have written a short program which illustrates the use of these two procedures by placing a number of blue fish on the screen against a yellow background.




```
10 REM Program FISH1
20 REM Version B1.0
30 REM Author Mike Williams
40 REM BEEBUG December 1983
50 :
100 MODE7:DIM fish$(5)
110 ON ERROR GOTO 220
120 FOR i=0 TO 23:VDU147,157,13,10:NE
XT i:VDU147,157
130 PRINT TAB(12,0);CHR$(132);"Flying
 Fish"
140 PROCdefinefish
150 FOR y=1 TO 19 STEP 6
160 IF (y-1)MOD12<>0 THEN PROCplacefi
sh(4,14,y) ELSE PROCplacefish(4,5,y):PR
OCplacefish(4,23,y)
170 NEXT y
180 :
190 REPEAT UNTIL FALSE
200 END
210 :
220 ON ERROR OFF:MODE7
230 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
240 END
```

There is a further point to notice here. The yellow background is produced, using the VDU command in line 120, by writing the appropriate control codes (147,156) down the left hand side of the screen. If you place such codes on the very bottom line of the display you must avoid following them with <return><linefeed> as this will cause the screen to scroll up one line. Here, I have dealt with lines 0 to 23 in a loop and then added the codes for the last line at the end. Problems can similarly arise if you try to print a character in the very last position on the bottom line, as the cursor automatically moves to the beginning of the next line down, and again the screen will scroll.

An interesting alternative technique for defining and displaying our fish is to treat the 5 rows of 12 characters as a text window. All the 60 characters that make up the fish are now assigned as a single character string to the variable fish$. By defining a text window of the right size and printing fish$ from the top left hand corner, all the characters of the fish are quickly placed in the correct positions. In order to avoid the scrolling problems described above the text window is defined to be one line lower than might appear necessary. The colour control codes are first placed in their right positions before displaying the fish and then the text window is restored to the default of the whole screen area.

```
 1000 DEF PROCdefinefish
 1010 fish$="":FOR i=1 TO 60
 1020 READ f%:fish$=fish$+CHR$(f%)
 1030 NEXT i
 1040 ENDPROC
 1050 :
 1060 DEF PROCplacefish(c,x,y)
 1070 LOCAL i,j
 1080 FOR i=0 TO 4:PRINT TAB(x-1,y+i);C
HR$(144+c);:NEXT i
 1090 VDU28,x,y+5,x+11,y,30:PRINT fish$;
 1100 VDU26
 1110 ENDPROC
 1120 :
 1130 DATA 160,224,254,255,255,252,176,
160,160,160,160,250
 1140 DATA 224,255,235,255,255,255,255,
244,160,160,248,255
 1150 DATA 254,255,255,255,255,255,255,
255,255,255,255,255
 1160 DATA 160,171,255,255,255,255,191,
163,160,160,171,255
 1170 DATA 171,239,255,255,175,161,160,
160,160,160,160,235
```

One consequence of this method, besides using fewer instructions and being quicker, is that any fish can be rapidly removed from the screen by redefining its text window and executing a CLS command. This is much simpler and quicker than replacing colour characters by conceal characters in the first version of this program. You might like to write a procedure called PROChidefish(x,y) to do this for each of the two versions of the fish program and compare them together.

Next month we will conclude this series on the Teletext Mode by presenting a set of procedures that will help you in designing Teletext displays. For complex graphics like the fish in this month's example, a good Teletext Editor offers a simple solution and BEEBUG is now able to offer one as part of the BEEBUG Software Library (see elsewhere in this issue).

POINTS ARISING

TAPE RECORDER REVIEW - BEEBUG Vol.2 No.5

Since our review we have had the opportunity of trying a second Data Recorder from Acorn and this has proved completely reliable in use. An almost identical tape recorder under the brand name Network (model NW900) is available from Argos and other stores at a price of approximately £22. This does not have a VU meter but does have a built in microphone for audio purposes.

LIGHT PEN - Vol.2 No.4

Enhancement:

In line 280 of the program, the offset variable, OS%, was set at 1542 (&606hex) with the program in mode 2. Mr G.P.Ball of Leicester has calculated the offset values in other modes as follows:

MODE	OFFSET
0, 1 & 2	1542 (&606hex)
3	2054 (&806hex)
4 & 5	2820 (&B04hex)
6	3076 (&c04hex)
7	10248 (&2808hex)

The formula for character resolution is S/C where 'S' is the number of characters per line for screen mode and 'C' is number of characters per line set by CRTC for mode.

Correction:

The parts list on page 20 incorrectly states the Maplin part number for the 18 SWG wire. It should read BL12N. We apologise for any inconveniance caused.

FIVE BOOKS FOR CHRISTMAS

Reviewed by Sheridan Williams

The BBC Micro Book - Basic, Sound & Graphics by J.McGregor & A.Watt, published by Addison Wesley at £7.95.
ISBN Ø-2Ø1-14Ø58-6
Cassette also available at £7.95.

For those just starting with the Beeb this book has an easy style. It covers Basic in a good manner using meaningful variable names right from the start.

This book, both text and programs, appears to me to have been produced using a daisy wheel printer. This is also true of the two following books. In this particular case unfortunately, the print wheel used did not have some of the characters used on the Beeb and the listing on page 153 contains some weird characters. The book includes several black and white screen photographs showing some of the results from the programs.

A major disappointment to me is the lack of a good description on files. I was just getting into their description when it stopped. Files need at least a couple of chapters.

The index and contents pages are generally good, although the index doesn't always send you to the correct page. There are some extra blank pages at the end of the book to make your own notes, a very useful feature.

Advanced Programming Techniques for the BBC Micro by J.McGregor & A.Watt, published by Addison Wesley at £7.95.
ISBN Ø-2Ø1-14Ø59-4
Cassette also available at £7.95.

This book is a followup from the one reviewed previously. It deserves the title "Advanced", but this should not put you off buying it. On the other hand you must have mastered most ordinary programming techniques before you read this book. It is a very readable book with several very interesting chapters.

There are so many good points it is difficult to pick a few to mention. However here goes - a good LOGO interpreter, and good chapters on recursion, sorting and searching. There is even a mention of PLOT 92 for filling.

Again I was very disappointed at the lack of a chapter on files. Surely in an Advanced book this is essential, especially as this topic was so poorly covered in "The BBC Micro Book". I can thoroughly recommend this book despite this fact.

BBC Micro Graphics and Sound by Steve Money, published by Granada at £6.95.
ISBN Ø-246-12156-4

I was not very impressed with this book. There are lots of niggling little points such as FOR loops for delays used instead of INKEY; poor print quality of program listings; use of the variable O; errors in Basic statements; use of crossed zero in the text sometimes, and sometimes not; not much use made of meaningful variable names.

This book assumes a knowledge of programming, and a familiarity with the Beeb, but for the user who has learnt to program in a structured way, the book will probably kill off these good habits.

For a book half devoted to graphics it has surprising omissions. For instance there is no mention of shifting of the graphics origin; no mention of PLOT parameters 72-79 or 88-95. There is no mention of downwards or sideways scrolling. On page 13 it implies that OS 1.Ø has been used to test programs rather than OS 1.2. In several places it mentions the use of the Epson MX8Ø, and yet doesn't give a screen dump.



BBC Basic by R.B.Coats, published by Edward Arnold at £5.95.
ISBN Ø-7131-3497-6

This is the cheapest of the 5 books reviewed here and represents good value for money. One of the best points with this book is the many example programs, some of them quite long, together with a description of how they were constructed. There is also a good chapter on "testing and debugging", both points which are severely neglected in most books.

There are one or two points that I was not too keen on. For example, Mr Coats states that BBC Basic provides a special variable for formatting numbers. However, he then states "this is beyond the scope of this book". I find this a strange statement when this is a very useful command. This book again has a poor section on files. Subroutines are mentioned before procedures, when one might question why on earth subroutines are mentioned at all.

Advanced Graphics with the BBC Model B Microcomputer by I.O.Angell & B.J.Jones, published by Macmillan at the price of £1Ø.
ISBN Ø-333-35052-9 (book)
ISBN Ø-333-35053-7 (cassette 1)
ISBN Ø-333-36141-5 (cassette 2)
Each cassette costs £9.ØØ.

This is a good book with a really superb two level index - a normal one and an index to all the procedures used in the book.

There is quite an emphasis on mathematics in this book, more so than in McGregor's Advanced book. This is necessary as the book concentrates on graphics whereas McGregor's book covers features other than graphics. This book covers graphics in great detail, and will provide you with lots to think about and try. However the price is pretty hefty at £1Ø. Some of the programs in the book are very long so it makes sense to supply these on cassette.

CONCLUSION

My favourite book is by Angell & Jones on Advanced Graphics. This is because of my interest in graphics, though you do need a mathematical background to make the most of this book. For the beginner I can recommend "The BBC Micro Book" as it starts from scratch and manages to cover so many topics. What a shame that not a single book covers the topic of files adequately. Some of the books scratch the surface, but stop almost as soon as they have begun.

Some of the publishers offer a cassette (or cassettes) to go with their book. This is a good solution, if a somewhat expensive one, to the chore of typing in long programs from book listings.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

INPUTLINE - P.G.Barnes

When entering data using INPUT and INPUTLINE, the TAB statement can be used to position the printed text and the input itself e.g.

```
INPUTLINE TAB(1Ø,5)"ENTER NAME"TAB(5,7) A$
```

Basic I & II
O.S. 1.2 only

SCREEN FREEZER (16k)

by David Graham

Have you ever wished you could take a pause in the middle of your favourite arcade game - either because you have been called away to the telephone in the middle of a high-scoring run, or simply so that you can examine the screen in peace while you work out a strategy for the next phase of the game. FREEZER allows you to do just that.

FREEZER is a short machine code routine that is loaded before a game or other program to provide a game or pause control. It works perfectly well on most of the arcade games that we have used it with - see table - and it works equally well with Basic or machine code programs.

Once FREEZER is activated, pressing the '@' key during the playing of most games will give a delay of approximately 30 secs. If your game uses any sound effects then these will also be 'frozen'. The program can be altered to give different delays. Once the delay time is up, the game resumes as normal, and the delay button may be pressed again to effect further delays.

Pressing Break disables the routine, but it may be re-enabled by CALL &A00. A beep indicates that it has been successfully called.

DETAILED INSTRUCTIONS

Type in the program, and save it on cassette or disc before running it. When you RUN it, you will be asked for an address. This may be hex or decimal, but hex values should be preceded by '&'. Generally speaking you should enter &A00 here, unless you wish to locate the code elsewhere (if you suspect that your arcade game uses this space). You should then see the assembly listing scroll on the screen, and a beep should be heard. The machine should now behave exactly as normal, until you press the '@' key. It should then hang up completely for around 30 secs.

Once the routine is working you should save a machine code copy. Type *SAVE FREEZER A00 A40. You can now reload the code by *RUN FREEZER - again a beep will indicate a successful execution.

Games successfully frozen:

Killer Gorilla	Program Power
Alien Destroyer	Beebugsoft
Swarmer	Beebugsoft
Snorter	Beebugsoft
Arcadians	Acornsoft
Planetoid	Acornsoft
Snapper	Acornsoft
Monsters	Acornsoft
Hopper	Acornsoft

Note that if one of the above games refuses to freeze, it may be that you have a different version of the game to us. Bug Blaster and Astrotracker refused to be frozen.

Once the code is in your machine, and enabled, you can load and run any game you wish. Remember though, to execute CALL &A00 after pressing Break at any time.

THEORY

The program is based on the general interrupt handler given last month, and works by enabling a key press event, and instead of processing the interrupt normally, the routine goes through a long timing loop if the special key is pressed. This effectively locks up the machine completely until the specified time has elapsed. During this time the keyboard is itself disabled, so no further key-conditional routines may be incorporated. To change the time, you may alter the value 200 in line 380. It may be anything from 1 to 255.

A more flexible pause control can be effected using a switch on the user port. To do this, change the event number to 4 (from 2) in line 110, and replace the key detection and loop routine with the following:

```
370       LDA #0
380       STA &FE62
390.LOOP  LDA &FE60
400       CMP #255
410       BNE LOOP
```

You will also need to delete lines 420 and 430 which are no longer needed.

Using this routine, if any of the 8 user port lines are brought to earth, the machine will freeze until all are restored high. This allows a push-button freeze-on freeze-off control, giving greater flexibility.

```
 10 REM Program FREEZER
 20 REM Version B0.4
 30 REM Author David Graham
 40 REM Based on a routine
 50 REM By Trevor Pullen
 60 REM BEEBUG December 1983
 70 REM Program subject to Copyright
 80 :
 90 MODE 7:ON ERROR GOTO 500
100 INPUT"ADDRESS (if hex, precede wi
th &)",addr$
110 PROCassemble(2)
120 CALL start
130 END
140 :
150 DEF PROCassemble(ECODE)
160 FOR PASS=0 TO 3 STEP 3
170 P%=EVAL(addr$)
180 [
190 OPT PASS
200 :
210 .start
220 CLD
230 LDA #7:JSR &FFEE
240 LDX #(ECODE)          ;event code
250 LDA #14
260 JSR &FFF4             ;enable event
270 LDA #(entry MOD 256)
280 STA &220              ;lo byte
290 LDA #(entry DIV 256)
300 STA &221              ;hi byte
310 RTS
320 BRK
330 :
340 .entry
350 CLD
360 PHA:TXA:PHA:TYA:PHA:PHP
370 CMP #64:BNE out
380 LDA #200:STA entry-1
390 .z LDY #255
400 .y LDX #255
410 .x DEX:BNE x
420 DEY:BNE y
430 DEC entry-1:BNE z
440 .out PLP:PLA:TAY:PLA:TAX:PLA
450 RTS
460 ]
470 NEXT PASS
480 ENDPROC
490 :
500 ON ERROR OFF:MODE 7
510 REPORT:PRINT" at line ";ERL
520 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

USEFUL KEY DEFINITION FOR WORDWISE - Paul Baron

This is a very useful key definition to allow the programmer to search for a specific piece of text , using Wordwise, with just one key press.

```
*KEY9 |!|O|[5S
```

This sends the cursor to the top of the file, out of edit mode to menu option 5 - search and replace, and selects 'selective' search. All you have to do is type in the string you wish to search for, and the string you want to replace it with. Alternatively, if you wish to find a certain piece of text, you can search for that text, and when prompted for a replace string just press Return. Then when Wordwise has found the text, if you press Escape twice it will leave the cursor at the required place.

UPPER/LOWER CASE CHARACTERS - N.Ahmon

When the Caps Lock light is off normal typing gives lower case characters, and pressing Shift with a key gives upper case, but when the light is on, normal typing and Shift-typing gives upper case. However holding down Shift and pressing Caps Lock, will allow you to type in upper case, until Shift is pressed with a key, when lower case is typed. This effect is cancelled if Caps Lock or Shift Lock are pressed. This is useful for entering lower case variable names within a program.

FITTING AN EXTERNAL SPEAKER TO THE BEEB

by Philip Le Grand

The small loud speaker fitted into the BBC micro does not really do justice to the range of sounds that the machine is capable of producing. This straightforward project shows you how to connect an external speaker to your Beeb at a very small cost (£1.20 approx. + cost of a speaker). You will find this vastly improves both the volume and quality of the notes produced by your Beeb and provides an excellent hardware project for those with limited experience in this area.

This article describes one easy method of connecting an external speaker to the Beeb using a jack plug and socket. The wiring is very simple and can be easily removed if you ever need to take out the Beeb's main board. In addition, the internal speaker still operates when the external speaker is unplugged. This project requires some simple soldering, but not to either of the two boards in the Beeb. You will, however, need to make a small hole in the case, to accommodate the jack socket.

The internal amplifier on the Beeb is capable of delivering 250mW of power, so there should be ample volume for an external speaker. You will also find that the 'quiet' buzz emitted from the internal speaker turns out louder with the external speaker, but you can always leave it unplugged when not in use.

LIST OF PARTS

COMPONENT	PAGE	ORDER No.	QTY	PRICE EACH
Minicon Latch Housing,2-way	149	HB59P	1	15p
Minicon Terminal	149	YW25C	2	3p
3.5mm socket	142	HF82D	1	14p
3.5mm jack plug	142	HF81C	1	24p
Single core screened cable	79	XR16S	1.5m	42p

Note: The pages listed are those of Maplin's 1983 catalogue, and the prices are from their current price list and include VAT. However, for orders less than £5, an extra 50p must be added for postage and packing. The parts may be obtained from:
Maplin Electronic Supplies Ltd., P.O. Box 3, Rayleigh, Essex, SS6 8LR.

CONSTRUCTION

All the components used are generally available from Maplin (and other stockists) and we have listed the complete set of parts needed. To fit the interface, proceed as follows:

1. Disconnect your micro from the mains. This is very important.

2. Remove the 4 screws securing the top half of the case.

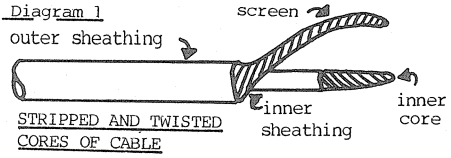
3. Remove the 2 or 3 screws securing the keyboard.

4. Carefully disconnect from the keyboard, the large ribbon cable joining the keyboard to the main board.

5. Remove the connector on PL15. This connector joins the internal speaker to the main board.

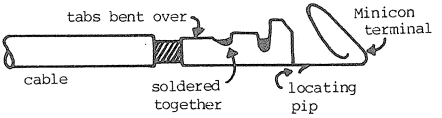
6. Take a 23cm lengh of screened cable, strip off about 1.5cm of the outer plastic, and twist the screening to form a single lead. Likewise, remove about 1cm of the plastic from the inner core and twist, so that the wire resembles diagram 1.

Diagram 1



7. Place the screening lead onto a Minicon connector, and bend the protruding metal tabs over with a pair of fine nosed pliers so that the wire is clamped. Then carefully solder the two together with a low power soldering iron (15W), to form a strong connection as in diagram 2. Now solder the other terminal to the central core in the same way.

Diagram 2 SIDE VIEW OF TERMINAL



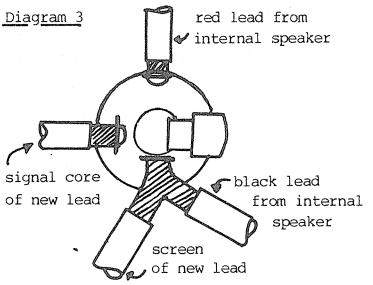
8. Now pick up your 2-way Minicon Latch Housing (it is the small box shaped unit), and notice that at one end there are two large holes and at the other, there are two small holes. The large holes will take the terminals you've soldered. On one side there are two rectangular holes - these accommodate the pips on the back of the terminals.

9. If you place the housing (as it is) onto PL15 with the pins of PL15 entering the small holes, then you should notice that there is only one correct way round with the two rectangular holes facing you, because there is a capacitor in the way. Now remove the housing, and insert the terminal on the screening into the right most large hole, ensuring the pip on the terminal fits into the rectangular hole. Repeat with the second terminal in the left hand hole.

10. Push the connector onto PL15 and carefully bend the screened cable over, so that it will not touch the keyboard when replaced. Wind the cable around the edge of the board, past IC17. Bend the cable up over the edge of the case by the power supply, and round into the well by the left side of the case. Now the cable is shaped, disconnect the plug before continuing.

11. Now cut off the plug already on the end of the leads from the internal speaker and strip the insulation from

Diagram 3



REAR VIEW OF JACK SOCKET, SHOWING CONNECTION DETAILS

the ends of the two wires. Strip the other end of the screened cable, in a mannner similar to (6) above.

12. Solder the screened cable and the original speaker leads to your 3.5mm jack socket as shown in diagram 3.

13. Carefully make a hole (ideally 6.35mm in diameter) through the case into the well, so that the jack socket is a good fit about half way along its length. Remove any bits of plastic remaining around the hole, as there is very little room to fit the socket. Unscrew the retaining ring on the socket and discard the washer underneath, since there is no extra room to fit this. Insert the socket into the hole and screw on the retaining nut - you may need to use pliers to tighten the nut. Don't over tighten, or you may break the thread on the socket.

14. Reconnect the keyboard, pushing any excess wire from the internal speaker, into the well, screw it in place and fix down the top of the computer.

15. Now you will need to make up a lead to connect a 3.5mm jack plug to the external speaker. Unscrew the cover on the 3.5mm jack plug, and solder the bared ends of the remaining length of screened cable to it, soldering the screen to the largest outer terminal, and the inner core to the inner terminal. Screw the cover back on to the plug.

16. Use a speaker with a power rating of a couple of Watts at least and

8 ohms impedance, otherwise over loading of the speaker will occur and the sound will be distorted. An old car radio speaker is ideal for the purpose. Our speaker was rated at 8W, and had an impedance of 8 ohms. It reproduced both the low and high frequency notes with a lot more feeling than the original speaker. Solder the other end of the cable to the speaker, ensuring the screen is soldered to the terminal marked with a '-' sign, or coloured black.

17. Finally plug the speaker into the socket, enter a program containing SOUND commands, and sit back and listen!

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

BIGGER BASIC PROGRAMS IN MODES 0,1 & 2 - J.Pike

You can 'steal' some of the display memory to increase the available RAM for Basic programs by preceding your program (as saved on tape) with:

```
1 L%=1:M%=1:*TV1
2 MODE M%:HIMEM=&3000+L%*&280
3 VDU 24,0;0;1279;1023-L%*32;
4 VDU 28,0,31,80/2^M%-1,L%
5 CHAIN""
```

In line 1, L% sets the number of lines of screen display used, where each line releases 640 bytes to Basic and M% sets the mode used. The garbage which will appear on the top line of the display when the main Basic program runs, is removed with *TV1. This only works on about the top 3 lines, before the display breaks up. If more than 3 lines are used (i.e. L%>3) then you must live with a mottled display at the top of the screen, but you can stop it "flashing" with *FX9,0. Line 2 sets the mode to 0,1 or 2, then resets HIMEM to the new top of Basic RAM. Lines 3 and 4 reset the graphics and text windows to protect the Basic RAM. Note that if VDU 24 or 28 are used in the main program,the size of the windows must not be increased above that set in lines 3 and 4.

SUMMARY OF START-UP OPTIONS - A.K.Bhanja

Break-Break	Hard reset (OS 0.1 only)
Break	Ordinary Reset
Ctrl-Break	Hard reset (1.2 OS only)
Shift-Break	Auto Boot Disc Filing System (DFS and 1.2 OS)
C-Break	select the cassette filing system (1.2 OS)
D-Break	select the disc filing system (1.2 OS)
N-Break	select Econet filing system (1.2 OS)
R-Break	select ROM filing system (1.2 OS)
Any key-Shift-Break	Auto Boot Cartridge ROM System (1.2 OS)

NULL FILENAMES - D.T.Goodwin

The DFS allows the creation of files with apparently no file name (they appear as a blank line in a directory catalogue - they may be detected using *INFO *.*). To delete these files some versions of the DFS allows the use of *DELETE "" but others insist on *DELETE " ".

TWO USEFUL LOCATIONS IN O.S. 1.2

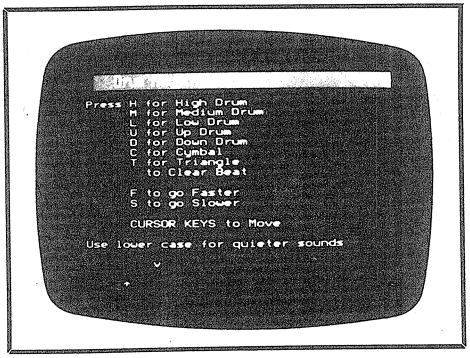
In operating system 1.2, the number of the screen mode currently being used is stored in location &355, which allows a mode independent program, for example a light pen procedure, to check which mode is being used by reading this location directly. Location &366 contains the character to be used as the output cursor in teletext mode. Try ?&366=42, in mode 7, and use the cursor keys. This should give an asterisk as a cursor.

Tested on O.S. 0·1 and 1·2 and on Basics I and II

THE PERCUSSION MACHINE (16k)

by Nick Day

We present here a fairly short program for generating percussion rhythms. Although the sounds produced are unsophisticated, the program is good fun to use, and is easy to modify for the purpose of experiment.



The Percussion Machine is an entertaining rhythm generator using a variety of percussion sounds. The top half of the screen displays a menu of all the sounds available, plus instructions. The lower part of the screen shows a marker scanning along a row of dots. A cursor, which is represented by the '^' character, can be moved by the cursor-left and cursor-right keys, and allows a drum noise to be entered at any point in the series. Any sound is entered by pressing a single key letter selected from the menu. Thus the program scans through the sounds presented on the screen making the appropriate sound for each one, but allows the list to be edited, by adding new sounds while the cursor is still scanning through those already set up.

You can also increase or decrease the tempo, again by single key strokes. For each instrument you can select a loud or a quiet sound.

The program can be used without any of the skills usually required for drumming, such as good timing and co-ordination, as the program does all the hard work for you. Experimenting with the various sounds available is very entertaining. The sounds include 5 types of drum, cymbal and triangle allowing great flexibility in output.

For the more adventurous the program is easily modified. It should be fairly simple to add more sounds, or to modify the existing ones. You may also feel that you could, by experimenting, improve upon the envelope definitions contained within the program. When adding more sounds remember to modify the command list in line 670 as well as the block of lines 300 to 440. You can also change the number of beats per cycle (currently 16) by changing the value in line 80. The structure of this program should make it easy to modify and as such could then be used as a basis for a much more powerful music development system.

```
  10 REM Program DRUMS
  20 REM Author Nick Day
  30 REM Version B1.3
  40 REM BEEBUG December 1983
  50 REM Program subject to copyright
  60:
  70 REM Initialise
  80 nsteps=16 : REM Cycle length
  90 REM this could be altered for
 100 REM other tempos.
 110 ON ERROR GOTO 1020
 120 DIM beat$(nsteps)
 130 MODE7
 140 VDU 23;10,32,0;0;0;
 150 PROCinit
 160:
 170 REM Main Loop
 180 REPEAT REM Cycle thru rhythm
 190 FOR timestep=1 TO nsteps
 200 PRINTTAB(editstep,21)SPC1
 210 editstep=(editstep+(INKEY-26)-(IN
KEY-122))
 220 IF editstep<1 THEN editstep=nstep
s
 230 IF editstep>nsteps THEN editstep=
1
 240 key=INKEY(0)
 250  IF key<>-1 THEN PROCsetchar(key)
 260 char$=beat$(timestep)
 270 PRINTTAB(editstep,21)"^"
```

```
  280 PRINTTAB((timestep-1)MOD nsteps+1
,19)" "
  290 PRINTTAB(timestep MOD nsteps+1,19
)"v"
  300 IF char$="." GOTO 450 : REM Sorry
, Edsger
  310 IF char$="L" THEN SOUND 0,1,6,1
  320 IF char$="l" THEN SOUND 0,2,6,1
  330 IF char$="M" THEN SOUND 0,1,5,1
  340 IF char$="m" THEN SOUND 0,2,5,1
  350 IF char$="H" THEN SOUND 0,1,4,1
  360 IF char$="h" THEN SOUND 0,2,4,1
  370 IF char$="D" THEN SOUND 1,3,255,1
:SOUND 0,1,7,1
  380 IF char$="d" THEN SOUND 1,3,255,1
:SOUND 0,2,7,1
  390 IF char$="U" THEN SOUND 1,4,150,1
:SOUND 0,1,7,1
  400 IF char$="u" THEN SOUND 1,4,150,1
:SOUND 0,2,7,1
  410 IF char$="C" THEN SOUND 0,2,4,1 :
 SOUND 2,1,197,1
  420 IF char$="c" THEN SOUND 0,2,4,1 :
 SOUND 2,2,197,1
  430 IF char$="T" THEN SOUND 3,1,245,1
  440 IF char$="t" THEN SOUND 3,2,245,1
  450 IF step<0 THEN step=0 : PRINT TAB
(30,20) CHR$(129)"Fastest"
  460 IF step>40 THEN step=40 : PRINT T
AB(30,20) CHR$(129)"Slowest"
  470 PROCwait
  480 NEXT timestep
  490 UNTIL FALSE
  500:
  510 DEF PROCwait
  520 REM Waits for next timestep
  530 REPEAT
  540 UNTIL TIME>nexttime
  550 nexttime=TIME+step
  560 ENDPROC
  570:
  580 DEF PROCclearbeat
  590 REM Clears the array beat$
  600 FOR I=1 TO nsteps
  610 beat$(I)="."
  620 NEXT
  630 ENDPROC
  640:
  650 DEF PROCsetchar(key)
  660 REM Takes action on input
  670 IF INSTR("HhMmLlUuDdCcTtFfSs.",CH
R$(key))=0 THEN ENDPROC
  680 IF key=ASC("S")ORkey=ASC("s")THEN
 step=step+1 : PRINT TAB(30,20)SPC8 : E
NDPROC
  690 IF key=ASC("F")ORkey=ASC("f")THEN
 step=step-1 : PRINT TAB(30,20)SPC8 : E
NDPROC
  700 beat$(editstep)=CHR$(key)
  710 PRINTTAB(0,20)CHR$(130); : FOR i=
1 TO nsteps : PRINTbeat$(i); : NEXT
  720 ENDPROC
  730:
  740 DEF PROCinit
  750 REM Initialises
  760 *FX4,1
  770 ENVELOPE 1,1,0,0,0,0,0,0,126,-10,
0,-1,126,100
  780 ENVELOPE 2,1,0,0,0,0,0,0,126,-20,
0,-1,126,60
  790 ENVELOPE 3,1,-2,0,0,255,0,0,0,0,0
,0,0,0
  800 ENVELOPE 4,1,2,0,0,255,0,0,0,0,0,
0,0,0
  810 PROCclearbeat
  820 step=10
  830 nexttime=TIME+step
  840 timestep=1
  850 editstep=1
  860 PRINT CHR$(148)CHR$(157)CHR$(131)
CHR$(141)"Drum Machine"
  870 PRINT CHR$(148)CHR$(157)CHR$(131)
CHR$(141)"Drum Machine"
  880 PRINT'"Press H for High Drum"
  890 PRINT"       M for Medium Drum"
  900 PRINT"       L for Low Drum"
  910 PRINT"       U for Up Drum"
  920 PRINT"       D for Down Drum"
  930 PRINT"       C for Cymbal"
  940 PRINT"       T for Triangle"
  950 PRINT"       . to Clear Beat"
  960 PRINT'"       F to go Faster"
  970 PRINT"       S to go Slower"
  980 PRINT'"       CURSOR KEYS to Move"
  990 PRINT'"Use lower case for quieter
 sounds"
 1000 PROCsetchar(ASC("."))
 1010 ENDPROC
 1020 ONERROR OFF
 1030 MODE7
 1040 IF ERR<>17 THEN REPORT:PRINT" at
 line ";ERL
 1050 *FX4
 1060 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CLG BUG - Dr.N.F.Kennedy

When using O.S. 1.2 and Basic I or Basic II, CLG does not return the graphics cursor to the origin, but leaves it where it was before the CLG instruction. O.S. 0.1 moved the cursor back to 0,0.

Basic I & II Disc Only

A LINK UTILITY FOR PROGRAMMERS (32k)

by Steve Hutt

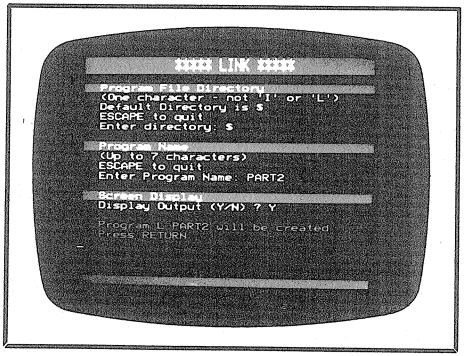
BBC Basic allows quite well structured programs to be written using procedures. Another useful approach is known as modular programming which entails dividing a program into small specific functions or modules. The code for each module is then written and tested independently. All of the program modules are then brought together or linked to form the complete program. This utility will carry out the linking process, including renumbering, and is a very useful aid indeed for program writers. The LINK utility is designed for use with disc systems and will run on both single and dual drive units. It incorporates extensive error checking.

A program module is like a short program that should be created, tested and saved on disc as for any other program. For the purposes of LINK all program modules should be saved on the same disc if possible and must be in directory 'I'. If necessary you can build up several discs of program modules provided they are all saved in directory 'I'. In this way you can build up a whole library of commonly used routines. When you want to write a program using the modular approach, you start by creating a root module (or program). At every point within the root program where a module is to be inserted, there should be a statement of the form:

REMINCLUDE<space><module name>

This statement should be entered directly following the Basic line number with no intervening spaces. There should be no space between the REM and the INCLUDE and there should be one, and only one space, between INCLUDE and the 'module-name'. The 'module-name' must be the file name of the program module as held on the disc. It should not include the directory 'I' as this is automatically assumed by the linking program. As typed, the 'module-name' should be 1 to 7 characters long and immediately followed by depressing the Return key. It is important to follow this format exactly, although the LINK utility does contain extensive error checking should you make a mistake.

The output program file that is produced by LINK will contain, in place of each REMINCLUDE statement, all of



the lines of code from the included module. If the module to be included cannot be found, a message is displayed on the screen, the REMINCLUDE statement is copied to the output program file and processing continues from the next statement. Note that programs may still be run even if they contain REMINCLUDE statements, as these statements are treated by Basic as comments.

Included module code may itself contain REMINCLUDE statements. Three nesting levels below the root module are allowed. Because only five channels are simultaneously available on the BBC micro, only five files may be open at any one time, one for the input program file, one for the output program file and three for nested 'include' files. This does not impose any limit on the number of modules that may be included in a program, only on the depth of nesting.

To use the LINK utility it should be loaded and run, and then the utility disc replaced by that holding the

program to be linked (the root module) and the modules themselves. The LINK utility will ask for the directory (which can be any character except 'I' or 'L') and the name of the Basic program to be linked. The default directory (normally $) is entered by pressing Return alone.

Once a program name has been entered the disc will be searched for that program. If it is not found then a new directory and program name will be requested. Pressing Escape at any time will exit from the utility. If the specified program is found, a new Basic program with the same name will be created in directory 'L'. This will be a copy of the root program, but incorporating the added modules.

During execution of LINK, the start and end of each included file is displayed on the screen. This is helpful because the nesting levels can be seen. At the start of the utility there is also an option for displaying the output code as files are linked. The code is displayed between the start and end lines thus showing the contents of each included module. Running the link utility without the display option,however, will greatly reduce the execution time. The line numbers displayed are those as held on the file. They will almost certainly be out of sequence but the program will be automatically renumbered later.It is not essential for each module to have unique line numbers, unless the files contain 'GOTO' type statements. It is best to number the lines in various modules above 20,000. Even with line numbers out of sequence, the renumbering will normally be successful.

One should never code GOTOs which jump outside of an included module. At some later stage the line number outside the module may be changed. In any case this utility should help you to write well structured code, thus avoiding GOTOs altogether.

When the linking ends, as well as residing on disc, the output program is left in the machine. The LINK utility includes an option that will automatically run the completed program once linking is complete.

If you have an Acorn DFS there is a limit of 31 files per side of a disc. Allowing for input and output program files, this only leaves space for 29 program modules. Should more be required they may be put on another disc. To use this, link the program using the first disc. Any 'includes' which are not resolved are copied to the output file. Insert the second disc and save the created program (which is still in the machine). Re-load the LINK utility and then re-insert the second disc of program modules. Re-running the link will now pick up any missing modules. This method obviously can be extended to as many discs as may be required and will also overcome the limitations on nesting described earlier.

The LINK utility is very useful for setting up.modules of code common to many programs. This is particularly so when writing assembler code. Most programs set up variables like OSWORD, OSBYTE, OSWRCH, etc. to hold the addresses of O.S. routines. These statements could be placed in a module called I.OSADDR. The start of each program would contain the code REMINCLUDE OSADDR.

In Basic, modules could consist of individual procedure definitions that are simply appended to the root program by the use of REMINCLUDE statements.

There are many other uses to which the LINK utility can be put. With a little thought it could even enable one to implement something along the lines of simple macros.

```
 10 REM Program LINK
 20 REM Version B1.9
 30 REM Author Stephen Hutt
 40 REM BEEBUG December 1983
 50 REM Program subject to Copyright
 60 :
100 ON ERROR GOTO 3200
110 PROCsetup
120 MODE7:VDU15
130 PROCfrontpage
140 PROCsetuplink
150 PROClink
```

```
  160 PROClastpage
  170 END
  180 :
 1000 DEF PROCsetup
 1010 DIM B%(255),C%(5),M%(12)
 1020 M%(3)=11
 1030 M%(4)=&F4:REM token for REM
 1040 M%(5)=ASC("I")
 1050 M%(6)=ASC("N"):M%(7)=ASC("C")
 1060 M%(8)=ASC("L"):M%(9)=ASC("U")
 1070 M%(10)=ASC("D"):M%(11)=ASC("E")
 1080 @%=&20005
 1090 ENDPROC
 1100 :
 1110 DEF PROCfrontpage
 1120 PROCmsgxy(0,0,"RNDY",STRING$(9,CH
R$32)+"***** LINK *****")
 1130 PROCmsgxy(0,1,"RNDY",STRING$(9,CH
R$32)+"***** LINK *****")
 1140 VDU28,0,24,39,2
 1150 PROCerror(STRING$(36,CHR$32))
 1160 VDU28,0,22,39,2
 1170 :
 1180 REPEAT
 1190 CLS:PROCmsgxy(0,1,"RNY","Program
File Directory")
 1200 PROCmsgxy(2,2,"C","(One character
 - not 'I' or 'L')")
 1210 PROCmsgxy(2,3,"C","Default Direct
ory is $")
 1220 PROCmsgxy(2,4,"C","ESCAPE to quit
"):flag%=FALSE
 1230 REPEAT
 1240 PROCmsgxy(2,5,"C","Enter director
y:"):PROCmsgxy(19,5,"Y","")
 1250 PROCblank(20,5):INPUTTAB(20,5)dir$
 1260 IF dir$="I" OR dir$="L" THEN PROC
error("Directory 'I' or 'L' not allowed
") ELSE IF LEN(dir$)>1 THEN PROCerror("
Too many characters") ELSE flag%=TRUE
 1270 IF LEN(dir$)=0 THEN dir$="$":PROC
msgxy(20,5,"","$")
 1280 UNTIL flag%
 1290 PROCerror(STRING$(36,CHR$32))
 1300 :
 1310 PROCmsgxy(0,7,"RNY","Program Name
"):PROCmsgxy(2,8,"C","(Up to 7 characte
rs)"):PROCmsgxy(2,9,"C","ESCAPE to quit
")
 1320 PROCmsgxy(2,10,"C","Enter Program
 Name:"):PROCmsgxy(22,10,"Y","")
 1330 flag%=FALSE
 1340 REPEAT
 1350 PROCblank(23,10):INPUTTAB(23,10)
prog$
 1360 IF LEN(prog$)>7 THEN PROCerror("P
rogram name too long") ELSE IF LEN(prog
$)<1 THEN PROCerror("No program name")
ELSE flag%=TRUE
 1370 UNTIL flag%
 1380 PROCerror(STRING$(36,CHR$32))
 1390 :
 1400 Lprog$="L."+prog$:P$=dir$+"."+pro
g$:CCT%=OPENUP(P$)
 1410 IF CCT%=0 THEN PROCerror("Program
 "+P$+" not found")
 1420 UNTIL CCT%:CLOSE#CCT%
 1430 PROCmsgxy(0,12,"RNY","Screen Disp
lay")
 1440 PROCmsgxy(2,13,"C","Display Outpu
t (Y/N) ?"):PROCmsgxy(25,13,"Y","")
 1450 REPEAT
 1460 PROCblank(26,13):INPUTTAB(26,13)
display$
 1470 UNTIL display$="Y"OR display$="N"
 1480 PROCmsgxy(2,15,"G","Program "+Lpr
og$+" will be created")
 1490 :
 1500 PROCmsgxy(2,16,"G","Press RETURN" )
 1510 CCT%=GET
 1520 ENDPROC
 1530 :
 1540 DEF PROCsetuplink
 1550 CCT%=1
 1560 C%(1)=OPENOUT(Lprog$)
 1570 BPUT#C%(1),&0D
 1580 BPUT#C%(1),&00:BPUT#C%(1),&01
 1590 BPUT#C%(1),&0B   :REM BYTE CT
 1600 BPUT#C%(1),&F4   :REM
 1610 BPUT#C%(1),ASC("L")
 1620 BPUT#C%(1),ASC("I")
 1630 BPUT#C%(1),ASC("N")
 1640 BPUT#C%(1),ASC("K")
 1650 BPUT#C%(1),ASC("E")
 1660 BPUT#C%(1),ASC("D")
 1670 BPUT#C%(1),&0D
 1680 ENDPROC
 1690 :
 1700 DEF PROClink
 1710 CLS:VDU14
 1720 PROCerror("Use SHIFT to continue")
 1730 PROCmsg("G","START OF FILE "+P$)
 1740 Z%=FNread(P$)
 1750 PROCmsg("G","  END OF FILE "+P$)
 1760 VDU15:PRINT:X%=POS:Y%=VPOS
 1770 BPUT#C%(1),&FF:CLOSE#C%(1)
 1780 PROCerror(STRING$(36,CHR$32))
 1790 ENDPROC
 1800 :
 1810 DEF PROClastpage
 1820 PROCmsgxy(X%+2,Y%,"C","Run Progra
m "+Lprog$+" (Y/N) ?"):PROCmsgxy(X%+32,
Y%,"Y","")
 1830 REPEAT
 1840 PROCblank(X%+33,Y%):INPUTTAB(X%+3
3,Y%) RN$
 1850 UNTIL RN$="Y" OR RN$="N"
 1860 VDU28,0,24,39,2:CLS
 1870 PROCmsg("LGD","LINK Completed")
 1880 PROCmsg("GD","LINK Completed")
```

```
1890 VDU26:PRINTTAB(0,5)
1900 PROCfinish
1910 ENDPROC
1920 :
1930 DEF PROCfinish
1940 HIMEM=HIMEM-10:$HIMEM=Lprog$:VDU21
1950 *K.0 LOAD $HIMEM|MRENUMBER|MSAVE
$HIMEM|MVDU6|M
1960 *K.1RUN|M
1970 *FX21,0
1980 *FX138,0,128
1990 IF RN$="Y" THEN *FX138,0,129
2000 ENDPROC
2010 :
2020 DEF FNread(filename$)
2030 LOCALI%,L%,X%,Y%
2040 L%=LEN(filename$):IF L%<3 OR L%>9
THEN PROCmsg("FY","INVALID FILE NAME "
+filename$):=FALSE
2050 IF CCT%>=5 THEN PROCmsg("FY","UNA
BLE TO OPEN FILE "+filename$):PROCmsg("
FY","MORE THAN 3 LEVELS OF NESTING"):=F
ALSE
2060 I%=OPENUP(filename$)
2070 IFI%=0 THEN PROCmsg("FY",filename
$+" NOT FOUND"):=FALSE
2080 IF EXT#I%>0 THEN X%=BGET#I%
2090 IF X%<>&0D THEN CLOSE#I%:PROCmsg(
"FY",filename$+" INVALID - NOT BASIC"):
PROCmsg("FY",filename$+" IGNORED"):=FAL
SE
2100 B%(1)=BGET#I%
2110 IFB%(1)=&FF OR EXT#I%<=2 THEN CLO
SE#I%:PROCmsg("FY",filename$+" IS EMPTY
"):=FALSE
2120 :
2130 CCT%=CCT%+1:C%(CCT%)=I%:B%(0)=I%
2140 REPEAT
2150 B%(2)=BGET#B%(0):B%(3)=BGET#B%(0)
2160 X%=3
2170 REPEAT
2180 X%=X%+1:B%(X%)=BGET#B%(0)
2190 UNTIL X%=B%(3)
2200 IF FNisitinclude=TRUE THEN PROCin
clude(B%(3)) ELSE PROCcopy(B%(3))
2210 B%(1)=BGET#B%(0)
2220 UNTIL B%(1)=&FF:CLOSE#B%(0)
2230 CCT%=CCT%-1:B%(0)=C%(CCT%)
2240 =TRUE
2250 :
2260 DEF FNisitinclude
2270 LOCAL X%,Y%
2280 IFB%(3)<=M%(3)THEN =FALSE
2290 Y%=TRUE
2300 FOR X%=4TOM%(3)
2310 IF B%(X%)<>M%(X%) THEN Y%=FALSE
2320 NEXT
2330 =Y%
2340 :
2350 DEF PROCcopy(L%)
2360 LOCAL I%
2370 FORI%=1TOL%
2380 BPUT#C%(1),B%(I%)
2390 NEXT
2400 IF display$="Y" THEN PROCdisplay(
L%)
2410 ENDPROC
2420 :
2430 DEF PROCdisplay(L%)
2440 LOCAL I%,QUOTE%
2450 PRINTTAB(0)B%(1)*&100+B%(2);
2460 QUOTE%=FALSE
2470 FORI%=4TOL%
2480 IF B%(I%)=ASC("""") THEN QUOTE%=N
OT QUOTE%
2490 IF B%(I%)=&0D THEN PRINT CHR$(&0D
) ELSE IF B%(I%)<&20 THEN PRINT".";
2500 IF QUOTE%=TRUE OR (B%(I%)>=&20 AN
D B%(I%)<=&7F) THEN PRINT CHR$(B%(I%));
2510 IF QUOTE%=FALSE AND B%(I%)>&7F AN
D B%(I%)<>&8D THEN PRINT FNkeyword(B%(I
%));
2520 IF QUOTE%=FALSE AND B%(I%)=&8D TH
EN PRINT FNlineno(B%(I%+1),B%(I%+2),B%(
I%+3));:I%=I%+4
2530 NEXT
2540 ENDPROC
2550 :
2560 DEF PROCinclude(L%)
2570 LOCAL I%,INC$:INC$=""
2580 IF B%(12)<>32 OR L%<14 THEN PROCm
sg("FY","INCLUDE NAME MISSING"):PROCcop
y(L%):ENDPROC
2590 FORI%=13TOL%-1
2600 IFB%(I%)>&7F THEN INC$=INC$+FNkey
word(B%(I%)) ELSE IF B%(I%)<>32 THEN IN
C$=INC$+CHR$(B%(I%))
2610 NEXT
2620 IF LEN(INC$)=0 THEN PROCmsg("FY",
"INCLUDE NAME MISSING"):PROCcopy(L%):EN
DPROC
2630 PROCmsg("C","INCLUDE "+INC$)
2640 IF NOT FNread("I."+INC$) THEN PRO
Ccopy(L%)
2650 PROCmsg("C","    END "+INC$)
2660 ENDPROC
2670 :
2680 DEF PROCmsg(cc$,msg$)
2690 LOCAL i%,c$
2700 FORi%=1 TO LEN(cc$)
2710 c$=MID$(cc$,i%,1)
2720 IF c$="C" THEN PRINTCHR$(134);
2730 IF c$="Y" THEN PRINTCHR$(131);
2740 IF c$="G" THEN PRINTCHR$(130);
2750 IF c$="R" THEN PRINTCHR$(129);
2760 IF c$="F" THEN PRINTCHR$(136);
2770 IF c$="D" THEN PRINTCHR$(141);
2780 IF c$="N" THEN PRINTCHR$(157);
2790 IF c$="L" THEN PRINTCHR$(13);CHR$
(10);
```

```
2800 NEXT:PRINT msg$
2810 ENDPROC
2820 :
2830 DEF PROCmsgxy(x,y,cc$,msg$)
2840 PRINTTAB(x,y);:PROCmsg(cc$,msg$)
2850 ENDPROC
2860 :
2870 DEF FNkeyword(K%)
2880 LOCAL I%,T%,KEY$
2890 T%=&806D
2900 REPEAT
2910 I%=T%
2920 REPEAT
2930 T%=T%+1
2940 UNTIL ?T%>&7F
2950 IF ?T%=K% THEN REPEAT:KEY$=KEY$+C
HR$(?I%):I%=I%+1:UNTIL I%=T%
2960 T%=T%+2
2970 UNTIL T%>=&8358 OR LEN(KEY$)>0
2980 =KEY$
2990 :
3000 DEF FNlineno(NO1%,NO2%,NO3%)
3010 LOCAL L%,L2%,L3%
3020 L2%=(NO1%AND&30)*4:L2%=L2%EOR NO2%
3030 L3%=(NO1%AND&04)*16:L3%=L3%EOR NO3%
3040 L%=L2%+L3%*256
3050 =STR$(L%)+CHR$32
3060 :
3070 DEF PROCerror(msg$)
3080 VDU28,0,24,39,2
3090 PROCmsgxy(0,21,"RN","")
3100 PROCmsgxy(2,21,"Y",msg$)
3110 PRINT STRING$(37-LEN(msg$),CHR$32
);
3120 VDU28,0,22,39,2
3130 ENDPROC
3140 :
3150 DEF PROCblank(x,y)
3160 PROCmsgxy(x,y,"",STRING$(40-x,CHR
$32))
3170 VDU28,0,22,39,2
3180 ENDPROC
3190 :
3200 ON ERROR OFF:CLOSE#0
3210 VDU28,0,24,39,2:CLS
3220 PROCmsg("LGD","LINK Terminated")
3230 PROCmsg("GD","LINK Terminated")
3240 VDU26:PRINTTAB(0,7);:REPORT:PRINT
" at line ";ERL
3250 END
```

Tested on O.S. 0.1 and 1.2 and on Basics I and II

MOVING CHEQUER BOARD (32k)

by D. D. Harriman

This month we bring you another short and fascinating graphics program by D.D.Harriman. This one is an excellent example of the animated graphics that can be achieved simply by switching colours.

This program presents the black and white squares of a chequer board in such a way that the squares appear both to get larger and move towards you as you watch. The program is essentially based on the use of the VDU19 command (see User Guide page 382). This is of the form:

VDU19,L,A,0,0,0

where L is the logical colour number (the range depends on the mode selected) and A is the actual colour (from 0 to 15). The program uses mode 2 which allows 16 logical colours. The chequer board is drawn initially using 14 of these colours which are then repeatedly changed between black and white to give the effects that you observe. The other two colours out of the 16 are used to provide white text and graphics against a black background in the top part of the screen.

PROGRAM DESCRIPTION

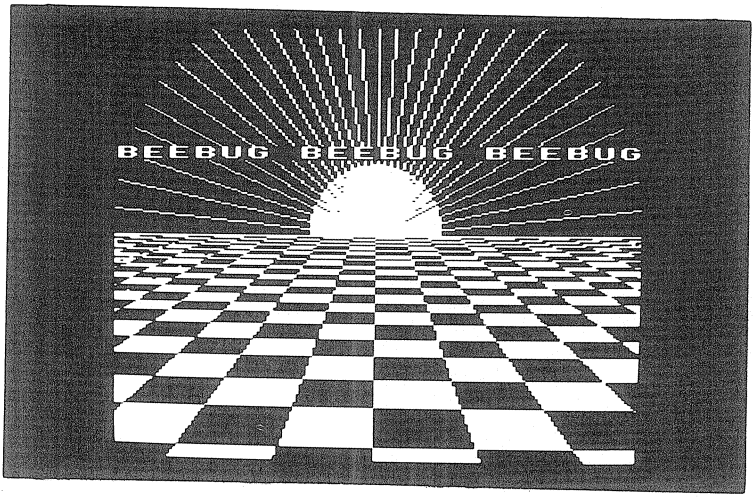
After some initial assignments the program starts by drawing bands of colour, decreasing in width, from the bottom of the screen up to approximately the halfway point (lines 150 to 190). The next routine (lines 210 to 270) uses exclusive OR plotting to draw perspective bands (each made of two triangles) across the horizontal lines. Each such band is four times as wide at the front as it is at the rear. Having done this the moving chequer board effect is produced by repeatedly executing the procedure PROCspeed which uses VDU19 to switch the colour assignments. We have embellished the area above the chequer board with some additional text and graphics (lines 290 to 380). You could easily define this area as a text and/or graphics window where other images could be produced.

SOME USEFUL VARIABLES

J% - determines the width and hence number of squares across the screen.

X% - sets the x co-ordinate for the centre of the circle.
Y% - controls the height of the chequer board on the screen.
Y%-Y - marks the actual height of the chequer board on the screen.
A$ - the text displayed on the screen at the end.

If you change the FOR statements at lines 410 and 440 to count in the reverse direction (up instead of down), you will find the chequer board moves away from you instead of towards you.



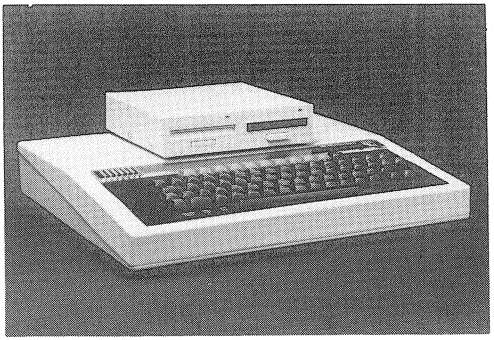

```
 10 REM Program CHEQUER
 20 REM Version B0.4
 30 REM Author Delos D.Harriman
 40 REM BEEBUG December 1983
 50 REM Program subject to Copyright
 60 :
100 MODE2:ON ERROR GOTO 540
110 J%=64:A$="BEEBUG "
120 VDU23;11,0;0;0;0
130 COLOUR7:COLOUR143:VDU19,15,0;0;12
140 X%=640:Y%=640:R=640:Y=Y%:C%=-1
150 FOR F=1 TO 6 STEP 0.03
160 C%=C%+1:C%=C%-(C%=7):IFC%=15:C%=0
170 GCOL0,128+C%:L=Y:Y=Y%/F
180 VDU24,0;Y%-L;1279;Y%-Y;:CLG
190 NEXT:VDU26:GCOL3,8
200 :
210 FOR F%=-640 TO 639 STEP J%
220 MOVE640+F%,Y%-Y:X1%=640+F%*4
230 MOVE X1%,0:F%=F%+J%:X2%=640+F%
240 Y2%=Y%-Y:PLOT85,X2%,Y2%
250 PLOT85,640+F%*4,0
260 MOVE X1%,0:DRAW X2%,Y2%
270 NEXT:GCOL0,7
280 :
290 FOR C=1 TO 4 STEP 3
300 FOR I=0 TO PI STEP PI/C/32
310 X1=X%+R*COS(I)/C:Y1=Y%+R*SIN(I)/C
320 MOVEX%,Y%-Y:DRAW X1,Y1-Y
330 NEXTI,C:PRINT TAB(0,9);
340 :
350 FORI%=1 TO 3:FOR F%=1 TO LEN(A$)
360 PRINT MID$(A$,F%,1);:PROCspeed
370 NEXTF%,I%
380 REPEAT:PROCspeed:UNTIL FALSE
390 :
400 DEF PROCspeed
410 FOR A%=6 TO 0 STEP -1
420 VDU19,A%,7;0;:VDU19,A%+8,0;0;
430 PROCW:NEXT
440 FOR A%=14 TO 8 STEP-1
450 VDU19,A%,7;0;:VDU19,A%-8,0;0;
460 PROCW:NEXT
470 ENDPROC
480 :
490 DEFPROCW
500 T%=TIME:REPEAT UNTIL TIME-T%>1
510 ENDPROC
520 :
530 ON ERROR OFF:MODE 7
540 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
550 END
```

DISC UNITS FOR THE BEEB

reviewed by Sheridan Williams

We review here three of the disc units that have recently become available. All are extremely compact, especially the new 3 inch drives from AMS, and at competitive prices. If you are thinking of purchasing a disc drive then this review may help you to make up your mind.

Model: AMS 3" Microdrives
Supplier: Advanced Memory Systems Ltd, Woodside Technology Centre, Green Lane, Appleton, Warrington, WA4 5NG.
Tel: 0925 62907.
Case size: single- (D)155x(W)95x(H)50mm
twin - (D)175x(W)190x(H)50mm
Price: £255 (Single), £399 (Double) (inc. VAT & delivery)



The drives used are Hitachi 305S microdrives. They use the power supply from the Beeb even for the twin drives. They are single sided drives with a capacity of 100k, however both sides of the disc can be used, effectively doubling their capacity to 200k.

As you can see from the dimensions above, these drives are exceedingly compact. I fell in love with them for this single reason. Besides the measurements the drives are pleasing aesthetically, discs are inserted horizontally, and the drives are side by side, rather than on top of one another. A very annoying feature is that the drive on the left is numbered 1 and the one on the right numbered 0. I am so used to reading from left to right that I immediately opened the case and reset the DIP switches (as explained in the manual) to change the numbering to the more sensible arrangement of left drive 0, right drive 1.

A very comprehensive, and well written reference manual is provided with the drives. It is printed using a matrix printer, and then photocopied, but nevertheless was very clear.

An interesting point with 3" discs is that the edge of the disc is still visible when inserted in the drive. There is enough room on the edge to write a title, which means that discs don't have to be removed to see what they are.

Another useful extra was an EPROM that contained two extra commands to enhance the BBC DFS - *FORMAT and *VERIFY. These are also supplied on the demonstration disc, but if you have a spare EPROM socket they are far more conveniently accessed from the operating system than having to find the demo disc each time they are required.

I found the drives very reliable. They have been in constant use now for over a month without any problems. In operation the head stepper motor was quiet, but the rotational motor was more noisy than I would expect.

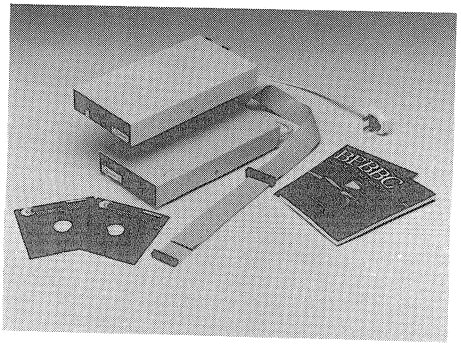
On the topic of compatability, I should point out that the manual makes it quite clear that one AMS drive can easily be connected together with an ordinary 5.25" drive, both capable of being used without unplugging. However for those with only 3" drives I have not seen any manufacturers selling software on 3" discs yet. Also there is no standard for micro floppy drives, and there are also 3.5" drives available to complicate matters further.

For those thinking about dual density controllers (we hope to be reviewing a couple next month) this would double the capacity from 100k to 200k per side, and the drives are quite capable of handling this.

I have two criticisms - one major and one minor. The minor one is that of very short data and mains leads. The leads were only 560mm long, which is barely long enough to place the drives in a convenient position. However, my major criticism with our twin drives was that of safety. Where the mains lead entered the drive casing there was no grommet or reinforcement of any kind. The mains lead had become twisted several times and was chafing dangerously on the unprotected metal. This is a major design fault and made the drives potentially very dangerous. I hope that AMS will pay particular attention to this in their future drive units.

Model: Cumana 5.25"
Supplier: Cumana, Unit 1, The Pines Trading Estate, Broad Street Guildford, Surrey, GU3 3BH. Tel (0483) 503121.
Case size: single-(D)295x(W)155x(H)52mm twin- (D)401x(W)204x(H)124mm
Type: Double-sided 80 track
Price: Single- £395.60, Double- £734.85 (inc VAT & delivery)

[Cumana normally supply through dealers. These are recommended retail prices and include manual and connecting cables.]



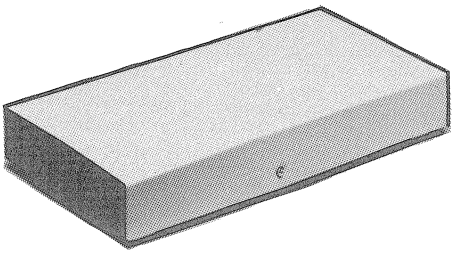
These Cumana units use Mitsubishi drives. We have had both a single and a twin Cumana disc unit on test for a while, although not as long as the AMS drives. The Cumanas have been reliable causing us no problems. They are very slim, but are quite deep, as the measurements show. This is because of their built-in power supply. The disc ejection mechanism is a nice feature, as not all 5.25" drives have this. Also the discs are inserted horizontally. Just as with the AMS drives, the drives are numbered strangely with the bottom drive as 0 and the top drive as 1.

A good feature is the length of the power lead and data cable, the latter being 1.25m long. This allows you to site the drives almost anywhere on your table top.

The drives had a slightly noisy stepper motor. The rotational motor however, was very quiet.

A manual can be purchased with the Cumana disc drives. It deserves much praise. It is well written, clear and informative.

Model: TEAC TCS 55ES
Supplier: Technomatic Ltd, 17 Burnley Road, London, NW10 1ED. Tel: 01-452 1500 and 01-450 6597
Case size: 275(D) x 150(W) x 52(H)mm
Type: Single-sided, 40/80 track switchable.
Price: £215 (+VAT)



These drives are very neat and attractive, and it is a shame that they don't quite fit on top of the Beeb, but overhang the back a bit; this is a minor point, however.

They are quiet in operation and come complete with a built-in 40/80 track switch included in the price. The discs are inserted horizontally, (which I prefer), and once inserted are locked in place with a small lever. The discs are not ejected when this lever is released, but are easy to remove anyway.

The drive uses the Beeb's power supply and can be sited quite conveniently thanks to 900mm long power and data cables.

CONCLUSION

I marginally prefer the TEAC disc drive because it is such good value for money, however all three makes can be recommended for a variety of reasons.

CHOICE OF DRIVE

One of the most common questions that we get is which disc system to choose. With all the different systems available the choice comes down to a choice of DFS, standard or double density controller, 40 or 80 track, single or double sided drives, single or dual drives.

The minimum (cheapest) system will be Standard density, 40\track, single sided, single drive, with a capacity of 100k.

The maximum (most expensive) system will be Double density, 80\track, double sided, dual drive, with a total capacity of 1600k.

There are dozens of combinations in between. For those upgrading to a disc system because they are bored with cassettes, the cheapest solution will probably suffice; though the Teac 40/80 switchable is also a very tempting option. Not until you graduate into handling large files, or you have hundreds of programs will you find the need for anything larger. Pound for pound, the cheapest way to obtain more capacity is to spend your money on a dual density controller (for example from Microware or LVL) rather than on larger capacity disc drives themselves. Also note that you can mix drives such as a single sided, 40 track drive with a double sided 80 track drive, or even a 5.25" drive with a 3" drive. This means that your initial choice won't go to waste if you need to expand later, making yet another reason to go for the double density controller. We hope to review double density controllers in our next issue.

NOTE: Technomatic have agreed to give BEEBUG members a discount of £10 on the TEAC disc drive reviewed above for approximately one month. Please contact Technomatic directly, and quote your BEEBUG membership number.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

*FX3,10 - C.W.Robertson

When using *FX3,10 to send text to the printer only, VDU1 becomes redundant. It is not necessary to have a special code to send characters to the printer only as any character can now be sent directly.

AUTO VERSION NUMBERING - J.P. Carnell

When devloping a program it makes sense to save each new version of it under a new filename.

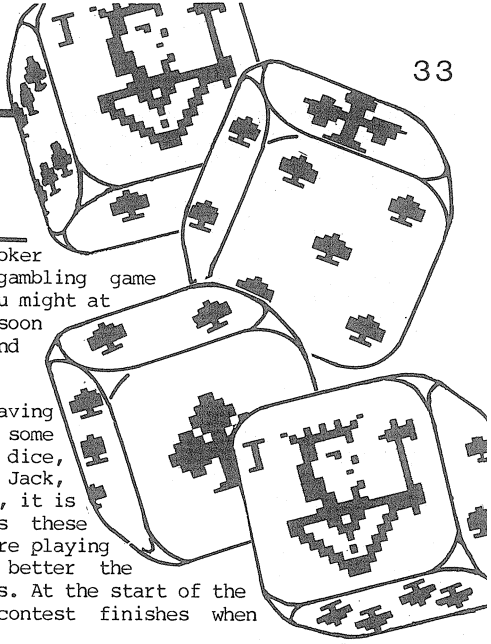
First, let Z%=0, then program f0: *KEY0,Z%=Z%+1|MSAVE"PROG"+STR$(Z%)|M

When you wish to save the new version press f0. This will save it as PROG1, PROG2 etc, with each subsequent depresssion of f0. The resident integer variables must be used for this purpose as they are retained during program editing. Take care also not to use the variable Z% anywhere in the program.

Basic I & II
O.S. 1.2 only

KILLER DICE (32k)

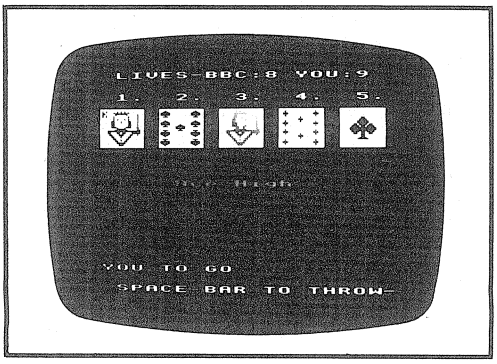
by D. Jackson



Why not treat yourself to our excellent poker dice game, called "Killer Dice"? This is a gambling game which involves more thought and skill than you might at first think. Like all true gamblers you will soon find yourself having just one more game, and one which you are bound to win this time!

For those of you who won't own up to having played with poker dice before, we start with some basic information. There are five identical dice, each with six faces representing the 9, 10, Jack, Queen, King and Ace. When the dice are thrown, it is the uppermost faces which count, and it is these which are displayed on the screen. You are playing against the computer, taking it in turns to better the score of each other until one of you loses. At the start of the game, both players have 9 lives and the the contest finishes when one of the players has no more lives left.

Except when you are throwing first at the start of a new game, you have the option of retaining any of the dice on the table and just throwing the remainder. This is the course of action to follow if some of the dice on the table are particularly high scoring. In this computer version, you can 'hold' any of the dice 1 to 5 by pressing any of the function keys f1 to f5 before pressing the space bar for your next go. If you make a mistake, pressing f0 clears any of the dice currently in a hold state and you can start again.



Different combinations of dice have different scoring values. Here are the winning combinations in order (highest value first):

1. 5 of a kind	6. 3 of a kind
2. 4 of a kind	7. Two pairs
3. Full House	8. One pair
4. High run	9. Ace high
5. Low run	

If a player throws 5 of a kind, his partner is given 5 throws to equal or better the hand. A full house is when a player gets 3 of a kind and a pair together.

As you get more experienced at playing Killer Dice, you will find yourself working out various strategies to defeat the computer. You will have to try hard to succeed and you will also need a certain amount of luck as well.

PROGRAM NOTES

The program is well structured and therefore relatively easy to follow, but you will need to take great care with the mass of data statements at the end. These are cunningly used to define the characters that represent each of the dice on the screen. The card number (9 to 13) indexes to a DATA statement which gives the colour followed by the 15 character definitions that are then poked in turn directly into the character buffer. Each character is dynamically defined as ASCII 224 and placed in a text window 3 wide and 5 deep to produce the dice face. Look at

the procedure PROCroll at line 2380 to see the detail of this. This is clever programming though unfortunately it won't work across the Tube.

```
   10 REM PROGRAM KILLER DICE
   20 REM VERSION B0.5
   30 REM AUTHOR  D.JACKSON
   40 REM BEEBUG  DECEMBER 1983
   60 :
   70 Prog$="KILLER DICE"
   80 ON ERROR GOTO3110
   90 :
 1000 MODE7:PROCintro
 1010 REPEAT:MODE2:PROCinitz
 1020 REPEAT:VDU17,3,31,0,26
 1030 IF T% PROCyou:ELSE PROCbbc
 1040 PROCthrow:PROCscore
 1050 IF B% T%=-T%-1:ELSE PROClose
 1060 UNTIL D%:PROCdead
 1070 UNTIL E%:MODE7:PROCend
 1080 END
 1090 :
 1100 DEFPROCintro
 1110 *FX225,128
 1120 DIMD(5),N(6),P(6),L(1)
 1130 Z%=RND(-TIME):T%=RND(2)-2
 1140 VDU31,11,8,141:PRINT Prog$
 1150 VDU31,11,9,141:PRINT Prog$
 1160 VDU31,5,17
 1170 PRINT"PRESS SPACE BAR TO CONTINUE"
 1180 VDU31,17,18
 1190 REPEAT:PROCinkey:UNTIL I%=32
 1200 ENDPROC
 1210 :
 1220 DEFPROCinitz
 1230 VDU23,1,0;0;0;0;
 1240 L(0)=9:L(1)=9
 1250 B%=0:F%=0:V%=1:E%=0:D%=0
 1260 VDU17,3,31,1,1
 1270 PRINT"LIVES-BBC:9 YOU:9"
 1280 VDU17,5,31,1,4
 1290 FORI%=1TO5:PRINT;I%;SPC(3);:NEXT
 1300 ENDPROC
 1310 :
 1320 DEFPROCyou
 1330 PRINT"YOU TO GO";
 1340 Z%=B%OR(-F%>0AND-F%<5)
 1350 IF Z% VDU17,1:PRINT"-INPUT HOLD"
 1360 VDU17,3,31,0,29
 1370 PRINT"-SPACE BAR TO THROW-"
 1380 REPEAT:PROCinkey:I%=I%-128
 1390 IF Z%:IF I%>0 AND I%<6:PROChold
 1400 IF I%=0:PROCcancel
 1410 UNTIL I%=-96
 1420 ENDPROC
 1430 :
 1440 DEFPROCbbc
 1450 PRINT"MY TURN":PROCdelay(3)
 1460 IF B%=0 AND F%=0:ENDPROC
 1470 X%=0:Y%=0
 1480 FORI%=6TOV%STEP-1
 1490 IF N(I%)>X%:X%=N(I%):Y%=I%
 1500 NEXT
 1510 IF X%=1 AND F%=0:ENDPROC
 1520 FORI%=1TO5
 1530 IF D(I%)=Y% PROChold
 1540 NEXT
 1550 ENDPROC
 1560 :
 1570 DEFPROCthrow
 1580 FORI%=1TO6:P(I%)=N(I%)ANDB%:NEXT
 1590 PROCdelay(2)
 1600 FORI%=1TO5
 1610 IF D(I%)<8 PROCdice:VDU12
 1620 NEXT
 1630 VDU28,0,31,19,15-(B%*2),12,26
 1640 FORI%=1TO5:Z%=D(I%)
 1650 IF Z%<8 PROCdice:PROCdelay(1):PRO
Croll
 1660 NEXT:VDU26:PROCcancel
 1670 ENDPROC
 1680 :
 1690 DEFPROCscore:PROCdelay(2)
 1700 VDU28,0,17,19,15,12,26
 1710 X%=0:Y%=0:Z%=0
 1720 FORI%=1TO6
 1730 IF P(I%)=0:X%=X%+1
 1740 IF N(I%)=0:Y%=Y%+1
 1750 IF N(I%)>Z%Z%=N(I%)
 1760 NEXT:X%=X%ANDB%
 1770 IF X%=1:IF P(1)=0 OR P(6)=0:X%=4
 1780 IF Y%=1:IF N(1)=0 OR N(6)=0:Y%=4:
Z%=5
 1790 IF F%=0 AND Y%=5:F%=-6:V%=D(5)
 1800 IF F% PROCfive:ENDPROC
 1810 PROCprompt:PROCdelay(1)
 1820 IF Y%>X%:B%=TRUE:ENDPROC
 1830 IF X%>Y%:B%=FALSE:ENDPROC
 1840 B%=1:X%=16
 1850 REPEAT:Y%=6
 1860 REPEAT
 1870 Z%=(P(Y%) OR N(Y%)) AND X%
 1880 IF Z%>0 AND N(Y%)>P(Y%) B%=TRUE
 1890 IF Z%>0 AND N(Y%)<P(Y%) B%=FALSE
 1900 Y%=Y%-1:UNTIL Y%=0 OR B%<1
 1910 X%=X%DIV2:UNTIL X%=0 OR B%<1
 1920 IF B%=1:B%=FALSE
 1930 ENDPROC
 1940 :
 1950 DEFPROClose
 1960 IF F% ENDPROC
 1970 PROCdelay(3):VDU17,1,31,2,17
 1980 IF T% PRINT"YOU";:SOUND0,-10,10,1
0 ELSE PRINT" I";:SOUND1,-10,101,8:SOUN
D1,-10,81,8
 1990 PRINT" LOSE A LIFE"
 2000 L(-T%)=L(-T%)-1
 2010 VDU17,3,31,(-T%*6)+11,1,48+L(-T%)
 2020 IF L(-T%)=0:D%=TRUE
 2030 ENDPROC
```

```
 2040 :
 2050 DEFPROCdead
 2060 VDU31,4,22,17,9
 2070 IF T% PRINT"YOU ARE";ELSE PRINT"
I AM";
 2080 PRINT" DEAD"
 2090 VDU17,3,31,0,28
 2100 PRINT"SPACE BAR TO REPLAY"'
 2110 PRINT"OR RETURN TO FINISH"
 2120 REPEAT:PROCinkey:UNTILI%=32ORI%=13
 2130 IF I%=13:E%=TRUE
 2140 ENDPROC
 2150 :
 2160 DEFPROCend
 2170 VDU12,17,3,31,5,12:PRINT"<BYEEEE";
 2180 *FX225
 2190 ENDPROC
 2200 :
 2210 DEFPROChold
 2220 IF F%=-5 ENDPROC
 2230 D(I%)=D(I%)+8
 2240 VDU17,1,31,(I%*4)-3,12,72
 2250 ENDPROC
 2260 :
 2270 DEFPROCcancel
 2280 FORI%=1TO5
 2290 D(I%)=D(I%)AND7
 2300 VDU31,(I%*4)-2,12,127
 2310 NEXT
 2320 ENDPROC
 2330 :
 2340 DEFPROCdice
 2350 Y%=(I%-1)*4:VDU28,Y%,11,Y%+2,6
 2360 ENDPROC
 2370 :
 2380 DEFPROCroll
 2390 SOUND1,-15,200,1
 2400 N(Z%)=N(Z%)DIV2:Z%=RND(6):D(I%)=Z%
 2410 IF N(Z%)=0 N(Z%)=1:ELSE N(Z%)=N(Z
%)*2
 2420 RESTORE (Z%*50)+2750:READC%
 2430 VDU17,C%,17,135
 2440 FORJ%=1TO15:READX%,Y%:!&C004=X%:!&
C00=Y%:VDU224:NEXT
 2450 VDU17,128
 2460 ENDPROC
 2470 :
 2480 DEFPROCfive
 2490 B%=0:Z%=-F%<6 AND Y%=5 AND D(5)>=
V%
 2500 IF Z% V%=1:F%=0:T%=-T%-1:ENDPROC
 2510 F%=F%+1:IF F%=0:V%=1:ENDPROC
 2520 IF F%=-5:T%=-T%-1
 2530 VDU17,2,31,0,15
 2540 PRINT"Five of Kind To Beat"'
 2550 PRINT"Or Equal In ";-F%;" Throws";
 2560 IF -F%=1:VDU127
 2570 ENDPROC
 2580 :
 2590 DEFPROCdelay(X%)
 2600 TIME=0:REPEAT:UNTIL TIME=X%*50
 2610 ENDPROC
 2620 :
 2630 DEFPROCinkey
 2640 *FX15,1
 2650 I%=INKEY(9)
 2660 ENDPROC
 2670 :
 2680 DEFPROCprompt
 2690 VDU17,2,31,4,15
 2700 IF Z%=1 AND Y%=1 PRINT" Ace High"
 2710 IF Y%=2 PRINT"One Pair"
 2720 IF Z%=2 AND Y%=3 PRINT"Two Pairs"
 2730 IF Z%=4 AND Y%=3 PRINT "3 of a Ki
nd"
 2740 IF Z%=5 AND N(6)=0 PRINT"Low Run"
 2750 IF Z%=5 AND N(1)=0 PRINT"High Run"
 2760 IF Z%=4 AND Y%=4 PRINT"Full House"
 2770 IF Z%=8 AND Y%=4 PRINT"4 of a Kin
d"
 2780 ENDPROC
 2790 :
 2800 DATA0,&107C7C7C,&38100000,0,0,&10
7C7C7C,&38100000
 2810 DATA&7C7C3810,&00000038,0,0,&7C7C
3810,&00000038
 2820 DATA&10380000,&0038107C,&0038107C
,&7C7C3810,&10380000,&0038107C
 2830 DATA&38000010,&387C7C7C,0,0,&3800
0010,&387C7C7C
 2840 DATA&00001038,&7C7C7C10,0,0,&0000
1038,&7C7C7C10
 2850 DATA1,&10387C38,&10100000,0,0,&10
387C38,&10100000
 2860 DATA&7C381010,&10,&101038,&7C3810
10,&7C381010,&10
 2870 DATA&38101000,&101038,0,0,&381010
00,&101038
 2880 DATA&10000000,&1010387C,&1010387C
,&38101000,&10000000,&1010387C
 2890 DATA&1010,&387C3810,0,0,&1010,&38
7C3810
 2900 DATA4,&20202021,&21720000,&C2C2FE
FF,&55540000,&1E0A0800,&800000
 2910 DATA&1010101,&1010060,&C0C48082,&
8288D8C0,&8080808,&80A1E3E
 2920 DATA&F0F0000,&10101,&FFFFFE64,&64
E6C2C0,&FCFC0C08,&8080808
 2930 DATA&1010303,&6060C0C,&93831111,&
38386C6C,&8080,&C8C87C7C
 2940 DATA0,0,&3838,&7C7CC6D6,0,0
 2950 DATA2,&20505051,&51220000,&8282FE
FF,&55540000,&3E1C0800,&800000
 2960 DATA&3030303,&3030111,&1111111,&4
56D0101,&9CAAAA88,&9CBE3636
 2970 DATA&F0F0000,&7070707,&FFFFEE44,&
83839329,&FCFC0C08,&C8C8C8DC
 2980 DATA&1010303,&6060C0C,&93831111,&
38386C6C,&8080,&C8C87C7C
 2990 DATA0,0,&3838,&7C7CC6D6,0,0
```

```
 3000 DATA1,&51506061,&51520000,&182FEF,&55540000,&8080800,&800000
 3010 DATA&3030303,&1010151,&29111111,&456D0101,&88888888,&8080808
 3020 DATA&F0F0000,&3030303,&FFFFFE7C,&C7833901,&FCFC0C08,&BE888888
 3030 DATA&1010303,&6060C0C,&93831111,&38386C6C,&8080,&C8C87C7C
 3040 DATA0,0,&3838,&7C7CC6D6,0,0
 3050 DATA0,0,0,0,0,0,0,0
 3060 DATA0,0,&7E7E7E3C,&3C181800,0,0
 3070 DATA&3070707,&3030101,&FFFFFFFF,&DBDBBDBD,&C0E0E0E0,&C0C08080
 3080 DATA0,&10103,&7E3C1818,&189999DB,0,&8080C0
 3090 DATA0,0,0,0,0,0
 3100 :
 3110 ON ERROR OFF
 3120 MODE7:IF ERR<>17 REPORT:PRINT" at line ";ERL
 3130 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

COLOURING VIEW - S.Fagg

VIEW normally uses white text on a black background. However, if you want green text, you can't use the VDU 19 command when in VIEW.

If you enter the following line:

```
*KEY 0 "|S|A|B|@|@|@|M"
```

pressing f0 changes the colour of the text. The text mode of VIEW disables the function keys, but stores them, so that they may be restored using *FX225,1.

COUNT ERRORS - J.G.Lusmore

When using the OSWRCH command in Assembler to produce some output, going back to using PRINT TAB statements in Basic will give incorrect results. This can be remedied by updating the zero page location of COUNT, which is &1E, using the following Assembler code:

```
CLC:LDA &1E:ADC #1:STA &1E
```

DETECTING THE SHIFT AND CONTROL KEYS - M.Hiseman & N.Lee

*FX202 can be used to control the Shift-Lock and Caps-Lock keys with bits 4 and 5 of the value passed in the X register. However, when used as an OSBYTE call with A=202, the following is true of the value returned in the X register:

Bit 6 set if CTRL-<any key> is pressed
Bit 5 set if Shift-Lock off
Bit 4 set if Caps-Lock off
Bit 3 set if Shift-<any key> is pressed

Bits 3 and 6 cannot detect Shift or Ctrl on their own, only when they are pressed in conjunction with another key.

O.S. 1.2 KEYBOARD SCAN - T.Green

Some programs written for O.S. 0.1 operating system fail to work on O.S. 1.2, because they look for keyboard input at location &D7 (215). On O.S. 1.2, the 'current key pressed' locations are &EC and &ED. The scan values have remained the same.

SPEEDING UP A/D CONVERSION - D.E.Susans

You can speed up the conversion of a channel by setting up the converter chip to 8 bit resolution. This is done by using *FX190,8 before any ADVAL calls are used. The time for a conversion is now 4ms instead of 10ms. The output reading has the same nominal 16 bit range as for the normal 12 bit operation but the least significant 8 bits are now random, so use either ADVAL(C) AND %FF00 or ADVAL(C) DIV 256 depending on the output range required (The value of C is the A/D channel number).

Normal 12 bit operation is restored with *FX190,0. Direct entry to the ADC control register using either *FX151,192 or ?&FEC0 does not work, as ADVAL resets the ADC using the *FX190 value before initiating a conversion.

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

TELETEXT AND THE BBC MICRO

Dear Sir,

The main reason for writing is to ask about Teletext adaptors etc. I must say that the idea of getting free programs from the T.V. sounds interesting but what do I need? If I have a Ceefax set, does it mean I can sit in front of the set and take down the listings by hand, switching backwards and forwards to make sure they are correct, ending up cross-eyed and cross, or is there an adaptor I can buy to couple to the existing Teletext set, that will cost considerably less than the Acorn teletext adaptor?

Anyway many thanks for your excellent magazine. I find it most useful. I am still very new at all this but nevertheless each passing day shows a little more light at the end of the tunnel. But sometimes I think that each week the tunnel gets a bit longer.

Peter MacDonald

Reply: Mr MacDonald is quite right. With a CEEFAX set, you can copy listings from the screen, but with great difficulty, not least because of the special teletext format. At present you will need to buy the Acorn Teletext adaptor to download programs and data directly into your micro. As the amount of software available in this way is likely to be small the cost of the adaptor is probably not justified for the general user. The facility is likely to be used by the BBC in conjunction with various educational programmes in the future, and thus this link might be useful to schools and colleges.

GRAPHICS IN WORDWISE

Dear Sir

I have tried unsuccessfully to put my monogram [attractively printed at the head of the original letter - Ed] into WORDWISE format. Please could you supply some hints, methods etc. so that I do not have to keep saving and loading intermediate data tapes? My printer is an Epson MX80 FT3.

A.J.Robinson

Reply: Using the embedded command OC enables any ASCII codes to be sent to the printer. For example, including OC27,75,5,0,255,255,255,255,255 as an embedded command will produce a block 8 dots high and 5 dots wide. The one problem that might arise is that a graphics character with ASCII code 13 is treated by WORDWISE as CR and, if you have set *FX6,0, will automatically cause a LF character to be sent to the printer as well, upsetting the graphics printing. WORDWISE normally uses the OSASCI routine for output. Changing the vector at OSASCI to point instead to the OSWRCH routine might solve this problem.

[We want POSTBAG to be a regular and lively feature of BEEBUG but we do need you to write to us with your comments, questions, ideas and criticisms. If this issue of the magazine has provoked you in any way then we would like to hear from you. Editor.]

DISILLUSIONS CURED

Dear Sir,

Thanks for your useful reply concerning the "Illusions" program in BEEBUG Vol.2 No.5. It now works perfectly with the excellent effects that I had hoped for. The fault had been mine - "typing mistakes". Congratulations to the author of the program, M.Inglis - perhaps in the future I may be able to understand every line of his program; at the moment I'm baffled! Meanwhile my 4th, 5th and 6th form physics pupils will be equally fascinated by it.

One small point - I couldn't get the little man to apear until I replaced (in line 270) the 189, 189 group by a 255, 255 group.

J.G.Conway

Reply: We're pleased that this program was well received. It just shows how easy it is to make a mistake when typing a program into a micro from a printed listing - and we have had similar queries from other members. There is no error in the original line 270 - can we suggest that another "typing mistake" was unknowingly corrected by the new version of this line?

Tested on O.S. 0.1 and 1.2 and on Basics I and II

GALACTIC INVASION (32K) by L. P. Baxter

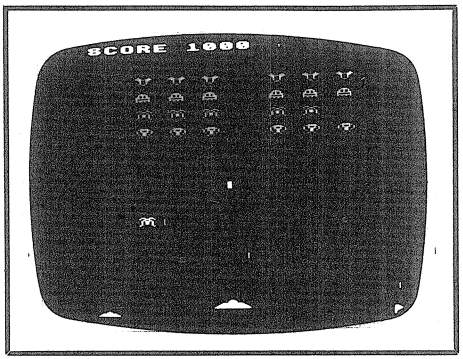
This program provides an impressive demonstration of how good arcade games can be when written in Basic on the BBC micro. The movement of the graphics appears quite smooth, the sound effects are excellent and the presentation is very professional.

The principle of this game is for you to shoot down the Galactic Invaders which appear as a massed force on the screen. As you might expect, they put up a good fight, swooping down one by one to attack you with purposeful but unpredictable trajectories. You move left and right across the bottom of the screen using the 'Z' and 'X' keys and fire back using the '.' key. When only one invader remains a new and even faster attack strategy is engaged.

The different invaders rate different scores if you destroy them, as is shown in the table at the start of the game. The highest scoring invaders vary randomly in value. If you successfully defend yourself against the first attack, you then face a second and further screens of invaders. You have three lives before your defences are exhausted. At any time during the display of the scoring or high score table, pressing the space bar will start the next game.

Like many good games programs this one is quite long and you must type it in carefully to avoid mistakes. Pay particular attention to the spaces in the program, though if you do get them wrong, it is only the initial displays which are likely to be affected. The mass of data statements at the end of the listing establishes the invaders' swoop patterns, so are not as critical as they might be, but errors will be thrown up if you do not have enough data. Please note that this program will not work if it is renumbered because of the use of variable RESTORE statements.

We did think of replacing our Galactic Invaders with characters of a more seasonal theme such as Cristmas puddings, crackers, Christmas trees etc. In the end we resisted the temptation, but if you feel like doing this then the four invader characters are defined in lines 880 to 910 and other characters like bullets and bombs in the lines following.

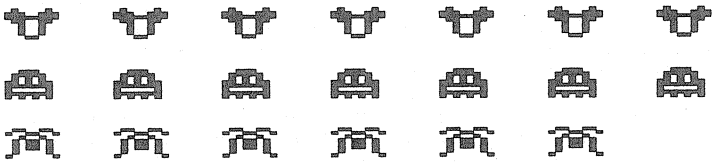



```
  10 REM Program GALACTIC INVASION
  20 REM Verson B0.9
  30 REM Author L.P.Baxter
  40 REM BEEBUG December 1983
  50 REM Program subject to Copyright
  60 :
 100 ON ERROR GOTO 20170
 110 PROCchars
 120 DIM A$(4),B$(4),X%(10),Y%(10),N$(
5),S%(5)
 130 FOR G%=1 TO 5:S%(G%)=6000-G%*1000
:N$(G%)="BEEBUG":NEXT
 140 REM==== COLD START ====
 150 MODE5:FORA%=1 TO 4:B$(A%)=CHR$(22
4+A%):NEXT
 160 VDU19,0,6,0,0,0:VDU19,3,4,0,0,0
 170 VDU19,2,5,0,0,0:PROCsheet1
 180 VOL%=-10:VOL1%=-10:ADD%=0:MIS%=0
 190 MIS$=CHR$230:FLAG$=CHR$231:FLAG10
$=CHR$238:NBASE$=CHR$235+CHR$236
 200 BASE$=CHR$32+CHR$232+CHR$233+CHR$
234+CHR$32:BASE%=0:NBASE%=2
 210 SHEET%=1:SHEET$=FLAG$:DIF%=30
 220 NMIS%=3:FREQ%=8:SHOT$=CHR$229
 230 REM==== WARM START ====
 240 SCORE%=0:MIS%=0:SHOT%=0:HSHOT%=0
 250 VDU 23;11,0;0;0;0
 260 CLS:COLOUR3:PRINT"SCORE":ZAP%=0
 270 NI%=28:REM  FLAG FOR LAST MAN(NO.
INV LEFT)
 280 FORA%=1 TO 4:A$(A%)=STRING$(6,CHR
$(224+A%)+CHR$32)+CHR$(224+A%):NEXT
 290 dive%=0:XDIVE%=0:YDIVE%=0:TYPE$="
":VP=0:HP=0:ALIEN%=2
 300 PROCinv:IF A%=0 THEN 300
```

```
  310 PRINTTAB(0,31);STRING$(NBASE%,NBA
SE$);TAB(19-LEN(SHEET$));SHEET$;
  320 DIR=0:PROCmove:new%=0:end%=0
  330 :
  340 REPEAT
  350 IF INKEY(-104)=-1 AND SHOT%=0 THE
N PROCfire
  360 IF INKEY(-98)=-1 THEN DIR=-1:PROC
move:ELSE IF INKEY(-67)=-1 THEN DIR=1:P
ROCmove
  370 PROCshot:PROCmis:PROCshot
  380 COLOUR2:PRINT TAB(6,0);SCORE%+ADD%
  390 IF SCORE%=7000 THEN PROCnewsheet:
new%=-1:GOTO 470
  400 IF ZAP%<>0 PROCzapped
  410 IF NBASE%=-1 PROCgamend:end%=-1:G
OTO 470
  420 REM NOW INVADERS TURN
  430 IF dive%=0 AND RND(20)=1 THEN PRO
Cinv:GOTO 450
  440 IF dive%=0 THEN 450 ELSE PROCdive
:GOTO 460
  450 IF RND(DIF%)=1 OR NI%=1 THEN PROC
startdive
  460 PROCshot
  470 UNTIL new% OR end%
  480 IF new% THEN 240 ELSE 150
  490 END
  500 :
  510 DEF PROCmove
  520 IF BASE%+DIR<0 OR BASE%+DIR>15 TH
EN ENDPROC
  530 BASE%=BASE%+DIR
  540 COLOUR2:PRINT TAB(BASE%,30);BASE$;
  550 ENDPROC
  560 :
  570 DEF PROCfire
  580 COLOUR3:SHOT%=BASE%+2:HSHOT%=29
  590 PRINTTAB(SHOT%,HSHOT%);SHOT$
  600 ENDPROC
  610 :
  620 DEF PROCshot
  630 IF SHOT%=0 THEN ENDPROC
  640 IF HSHOT%=4 THEN PRINT TAB(SHOT%,
HSHOT%);SPC1:SHOT%=0:HSHOT%=0:ENDPROC
  650 HSHOT%=HSHOT%-1:IF ?(SHOT%*16+HSH
OT%*32+2+HIMEM)<>0 THEN 670
  660 COLOUR2:PRINTTAB(SHOT%,HSHOT%+1);
SPC1;CHR$11;CHR$8;SHOT$:ENDPROC
  670 PRINTTAB(SHOT%,HSHOT%+1);SPC1;CHR
$11;CHR$8;"*";CHR$8;
  680 IF XDIVE%=POS AND YDIVE%=VPOS THE
N 690 ELSE 730
  690 dive%=0:A$(VP)=LEFT$(A$(VP),HP*2-
2)+CHR$32+MID$(A$(VP),HP*2)
  700 SCORE%=SCORE%+500-100*VP:VDU 32:I
F VP<>1 THEN ADD%=ADD%+500-100*VP:GOTO
760
  710 S%=RND(10)*100:ADD%=ADD%+S%:PRINT
 CHR$8;CHR$8;S%+400;
  720 PROCnoise2:VDU8,8,8,8,32,32,32,32
:GOTO770
  730 SCORE%=SCORE%+500-FNY(VPOS)*100
  740 A%=FNY(VPOS):B%=FNX(POS)*2-1:A$(A
%)=LEFT$(A$(A%),B%-1)+CHR$32+MID$(A$(A%
),B%+1)
  750 VDU 32:IF dive%=1 THEN PROCdive
  760 PROCnoise
  770 SHOT%=0:HSHOT%=0:NI%=NI%-1
  780 IF NI%=1 THEN DIF%=2:NMIS%=0
  790 ENDPROC
  800 :
  810 DEF PROCinv
  820 COLOUR1
  830 A%=RND(3)-2:IF A%=0 THEN ENDPROC
  840 IF ALIEN%+A%<1 OR ALIEN%+A%>4 THEN
 ALIEN%=ALIEN%-A% ELSE ALIEN%=ALIEN%+A%
  850 PRINT TAB(ALIEN%,4);SPC1;A$(1);SP
C1''TAB(ALIEN%);SPC1;A$(2);SPC1''TAB(AL
IEN%);SPC1;A$(3);SPC1''TAB(ALIEN%);SPC1
;A$(4);SPC1:ENDPROC
  860 :
  870 DEF PROCchars
  880 VDU 23,225,66,231,255,102,36,36,3
6,24
  890 VDU 23,226,56,124,84,214,254,130,
254,170
  900 VDU 23,227,102,153,36,90,90,90,66
,129
  910 VDU 23,228,60,126,165,189,153,90,
24,36
  920 VDU 23,229,0,24,24,24,24,24,24,0
  930 VDU 23,230,0,8,8,8,8,8,8,0
  940 VDU 23,231,48,56,60,56,48,32,32,32
  950 VDU 23,232,0,0,0,1,3,7,15,31
  960 VDU 23,233,24,60,60,255,255,255,2
55,255
  970 VDU 23,234,0,0,0,128,192,224,240,
248
  980 VDU 23,235,0,0,0,1,3,15,31,63
  990 VDU 23,236,0,0,0,0,128,224,240,248
 1000 VDU 23,237,0,0,0,255,0,255,0,0,0
 1010 VDU 23,238,176,184,188,184,176,16
0,160,160
 1020 ENDPROC
 1030 :
 1040 DEF PROCdive
 1050 IF NI%=1 THEN SOUND 1,VOL%,30,1:S
OUND 1,VOL%,10,1:GOTO 1070
 1060 SOUND &0011,VOL%,SOU%,5:SOU%=SOU%-1
 1070 IF XDIVE%<0 OR XDIVE%>19 THEN 1090
 1080 PRINT TAB(XDIVE%,YDIVE%);SPC1;:IF
 MIS%<NMIS% AND RND(FREQ%)=1 AND YDIVE%
<28 THEN MIS%=MIS%+1:X%(MIS%)=XDIVE%:Y%
(MIS%)=YDIVE%
 1090 IF YDIVE%>29 THEN 1120
 1100 READ A%,B%:XDIVE%=XDIVE%+A%:YDIVE
%=YDIVE%+B%:IF XDIVE%>=0 AND XDIVE%<20
THEN PRINT TAB(XDIVE%,YDIVE%);TYPE$;
 1110 ENDPROC
```

```
 1120 IF NI%=1 THEN 1130 ELSE IF RND(5)
=1 THEN PROCinv ELSE PRINT TAB(FNX1(HP)
,FNY1(VP));TYPE$
 1130 MIS%=0:dive%=0
 1140 IF XDIVE%>BASE% AND XDIVE%<BASE%+
4 THEN ZAP%=1
 1150 ENDPROC
 1160 :
 1170 DEF PROCstartdive
 1180 RESTORE (RND(3)-1)*20+11000
 1190 VP=RND(4):IF INSTR(A$(VP),B$(VP))
=0 THEN 1190
 1200 IF RND(2)=1 THEN 1220 ELSE A%=15
 1210 A%=A%-2:IF MID$(A$(VP),A%,1)=CHR$
32THEN 1210 ELSE HP=(A%+1)/2:GOTO 1240
 1220 HP=(INSTR(A$(VP),B$(VP))+1)/2
 1230 RESTORE (RND(3)-1)*20+10000
 1240 TYPE$=B$(VP)
 1250 IF NI%=1 THEN RESTORE (10060+(RND
(2)-1)*1000):PRINT TAB(FNX1(HP),FNY1(VP
));SPC1;:HP=3
 1260 XDIVE%=FNX1(HP):YDIVE%=FNY1(VP)
 1270 SOU%=200:dive%=1:ENDPROC
 1280 :
 1290 DEF FNX(X)=(X-ALIEN%+1)/2
 1300 DEF FNY(Y)=(Y-2)/2
 1310 DEF FNX1(X)=X*2+ALIEN%-1
 1320 DEF FNY1(Y)=2+Y*2
 1330 :
 1340 DEF PROCmis
 1350 IFMIS%=0 THEN ENDPROC
 1360 FORA%=1TOMIS%:IF X%(A%)=0 AND Y%(
A%)=0 THEN NEXT:ENDPROC
 1370 PRINTTAB(X%(A%),Y%(A%));SPC1;:Y%(
A%)=Y%(A%)+1:IF Y%(A%)<30 THEN 1380 ELS
E C%=X%(A%):X%(A%)=0:Y%(A%)=0:IF C%<BAS
E%+1 OR C%>BASE%+3 THEN 1390 ELSE ZAP%=
1:GOTO 1390
 1380 COLOUR1:PRINTTAB(X%(A%),Y%(A%));M
IS$;
 1390 NEXT:PROCshot:ENDPROC
 1400 :
 1410 DEF PROCzapped
 1420 NBASE%=NBASE%-1
 1430 COLOUR2:PRINT TAB(BASE%,30);BASE$;
 1440 PROCexplo:IF NBASE%=-1 ENDPROC
 1450 CLS:COLOUR3:PRINT"SCORE"
 1460 COLOUR2:PRINT TAB(0,31);STRING$(N
BASE%,NBASE$);SPC2;TAB(19-LEN(SHEET$));
SHEET$;TAB(0,30);BASE$
 1470 ZAP%=0:dive%=0:XDIVE%=0:YDIVE%=0:
MIS%=0:HP=0:VP=0:BASE%=0:HSHOT%=0:SHOT%=0
 1480 REPEAT:PROCinv:UNTIL A%<>0:ENDPROC
 1490 :
 1500 DEF PROCgamend
 1510 FOR A=1 TO 7000:NEXT
 1520 SCORE%=SCORE%+ADD%:IF SCORE%<=S%(
5) THEN 1690
 1530 REM MADE IT INTO HIGH SCORE TABLE
 1540 FOR A%=5 TO 1 STEP -1
 1550 IF S%(A%)<SCORE% THEN C%=A%
 1560 NEXT:VDU22,7
 1570 FORF%=1 TO 2:PRINTTAB(0,F%);CHR$1
41;CHR$130;"Galactic Invasion":NEXT
 1580 PRINT'CHR$134"Your score of"CHR$1
31;SCORE%;CHR$134"is enough to rank"
 1590 PRINT TAB(0,5)CHR$133;C%;MID$("st
ndrdthth",C%*2-1,2);CHR$134"in the high
 score table."
 1600 PRINT TAB(0,8);CHR$134"Please ent
er your name in not more than"CHR$136;C
HR$134"ten letters."
 1610  PRINT TAB(13,15);CHR$134">";SPC(
10);"<";SPC(13)
 1620 PRINT TAB(15,15);:*FX 15,1
 1630 INPUT ""A$
 1640 IF LEN(A$)>10 THEN 1610
 1650 IF C%=5 THEN 1680
 1660 FOR A%=4 TO C% STEP -1
 1670 S%(A%+1)=S%(A%):N$(A%+1)=N$(A%):N
EXT
 1680 S%(C%)=SCORE%:N$(C%)=A$
 1690 ENDPROC
 1700 :
 1710 DEF PROCnewsheet
 1720 ADD%=ADD%+SCORE%:SHEET%=SHEET%+1
 1730 SHEET$=STRING$(SHEET% MOD 10,FLAG
$)+STRING$(SHEET% DIV 10,FLAG10$)
 1740 NMIS%=3+INT(SHEET%/2):IF NMIS%>8
THEN NMIS%=8
 1750 DIF%=DIF%-5:IF DIF%<5 THEN DIF%=2
 1760 FREQ%=11-NMIS%
 1770 FOR A=1 TO 2500:NEXT:ENDPROC
 1780 :
 1790 REM DISPLAY SHEETS
 1800 DEF PROCsheet1
 1810 REPEAT:VDU 23;11,0;0;0;0;:*FX 15,1
 1820 KE%=0:PROChitable:IF KE%=0 THEN E
NDPROC
 1830 CLS:C%=20:*FX15,1
 1840 A$=" GALACTIC INVASION":PROCintro
(1)
 1850 A$=" "+STRING$(17,CHR$237):PROCin
tro(2)
 1860 A$="400  "+B$(1)+"   ?":PROCintro
(8)
 1870 A$="300  "+B$(2)+"  600":PROCintr
o(12)
 1880 A$="200  "+B$(3)+"  400":PROCintr
o(16)
 1890 A$="100  "+B$(4)+"  200":PROCintr
o(20)
 1900 A$="Z KEY  LEFT":PROCintro(26)
 1910 A$="X KEY  RIGHT":PROCintro(28)
 1920 A$=". KEY  FIRE":PROCintro(30)
 1930 IF INKEY$(1000)<>""THEN ENDPROC
 1940 UNTIL FALSE
 1950 :
 1960  DEF PROCintro(A%)
 1970 IF A%=1 OR A%=2 GOTO1990
 1980 A$=STRING$(4,CHR$32)+A$
```

```
 1990 FOR B%=15 TO 1 STEP -1:A=SIN(97.6
):B$=MID$(A$,B%):PRINT TAB(0,A%);B$;:NE
XT:ENDPROC
 2000 :
 2010  DEF PROChitable
 2020 CLS:A%=6
 2030 A%=A%-1
 2040 PRINT TAB(0,0);A%;". ";S%(A%);TAB
(10,0);N$(A%);CHR$30;
 2050 B%=0
 2060 B%=B%+1
 2070 VDU 11:IF INKEY(10)<>-1 THEN A%=0
:B%=5:ENDPROC
 2080 IF B%<4 THEN 2060
 2090 IF A%>1 THEN 2030
 2100 VDU 11,11,11:PRINT TAB(3,2)STRING
$(14,CHR$237);CHR$11;TAB(3,1)"HIGHEST S
CORES"
 2110 IF INKEY(1000)<>-1 THEN ENDPROC
 2120 KE%=1:ENDPROC
 2130 :
 2140 REM  EXPLOSION
 2150  DEF PROCexplo
 2160 FOR A%=1 TO 50
 2170 GCOL RND(4)-1,RND(3)
 2180 SOUND 1,0,RND(50)+105,1:SOUND 0,V
OL%,7,1
 2190 MOVE (BASE%+3)*64-32,32
 2200 DRAW RND(144)-96+(BASE%+3)*64, RN
D(144)+32
 2210 NEXT
 2220 FOR A=1 TO 5000:NEXT:ENDPROC
 9999 REM DATA FOR DIVES
10000 DATA -1,1,-1,1,-1,1,0,1,0,1,1,1,1
,1,1,1,1,1,1,1,1,0,1,0,1,1,1,0,1,0,1,1,
1,0,1,1,1,0,1,1,1,0,1,1,0,1,0,1,-1,1,-1
,0,-1,1,-1,0,-1,1,-1,1,-1,1,-1,1,-1,0,-
1,1,-1,1,-1,1,-1,1,-1,1
10020 DATA -1,1,-1,1,-1,1,0,1,0,1,1,1,1
,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,1,
1,0,1,0,1,0,1,1,1,0,1,1,0,1,-1,1,-1,0,-
1,0,-1,1,-1,0,-1,0,-1,1,-1,0,-1,0,-1,0,
-1,0,-1,1,0,1,1,1,1,1,1,1,1,1,1,0,1,1,1
,0,1,1,1,1,1,1,1,1
10040 DATA -1,1,-1,1,-1,1,0,1,0,1,1,1,1
,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,1,
1,0,1,0,1,0,1,1,1,0,1,1,0,1,-1,1,-1,0,-
1,0,-1,1,-1,0,-1,0,-1,1,-1,0,-1,0,-1,0,
-1,0,-1,1, 0,1,1,1,-1,1,-1,1,-1,1,-1,1,
-1,1,0,1,1,1,1,1
10060 DATA 0,1,0,1,0,1,0,1,0,1,0,1,0,1,
0,1,0,1,0,1,-1,1,-1,1,-1,0,-1,1,1,1,1,0
, 1,1,1,0,1,0,1,0,1,0,1,0,1,1,1,1,-1,1,-1,1
,-1,0,-1,0,-1,0,-1,0,-1,-1,-1,-1,0,-1,0,0,1,
-1,1,-1,1,0,0,0,1,0,0,0,0,1,0,1,1,1,1,1,0,1
,0,0,-1,1,-1,1
10070 DATA-1,1,-1,1,-1,1,-1,1,-1,1,-1,1,-1,1
,-1,1,-1,1,-1,1
11000 DATA 1,1,1,1,1,1,0,1,0,1,-1,1,-1,
1,-1,1,-1,1,-1,1,-1,0,-1,0,-1,1,-1,0,-1
,0,-1,1,-1,0,-1,1,-1,0,-1,1,-1,0,-1,1,0
,1,0,1,1,1,1,0,1,1,1,0,1,1,1,1,1,1,1,1,
1,0,1,1,1,1,1,1,1,1,1,1
11020 DATA 1,1,1,1,1,1,0,1,0,1,-1,1,-1,
1,-1,1,-1,1,-1,1,-1,0,-1,0,-1,0,-1,0,-1
,0,-1,1,-1,0,-1,0,-1,0,-1,1,-1,0,-1,1,0
,1,1,1,1,0,1,0,1,1,1,0,1,0,1,1,1,0,1,0,
1,0,1,0,1,1,0,1,-1,1,-1,1,-1,1,-1,1,-1,
0,-1,1,-1,0,-1,1,-1,1,-1,1,-1,1,-1,1
11040 DATA 1,1,1,1,1,1,0,1,0,1,-1,1,-1,
1,-1,1,-1,1,-1,1,-1,0,-1,0,-1,0,-1,0,-1
,0,-1,1,-1,0,-1,0,-1,1,-1,0,-1,1,0
,1,1,1,1,0,1,0,1,1,1,0,1,0,1,1,1,0,1,0,
1,0,1,0,1,1,0,1,-1,1,1,1,1,1,1,1,1,1,1,
1,0,1,-1,1,-1,1
11060 DATA 0,1,0,1,0,1,0,1,0,1,0,1,0,1,
0,1,0,1,0,1,1,1,1,1,1,0,1,1,1,-1,1,-1,0,-
1,1,-1,0,-1,0,-1,0,-1,0,-1,1,-1,1,1,1,1
,1,1,0,1,0,1,0,1,-1,1,-1,0,-1,0,0,-1,-1
,-1,-1,-1,0,0,0,-1,0,0,0,-1,0,-1,1,-1,1
,0,1,0,0,1,1,1,1
11070 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1
20000  DEF PROCnoise
20010 A%=187:B%=179
20020 SOUND 2,VOL%,109,2
20030 SOUND 2,VOL%,121,1
20040 SOUND 2,VOL1%,B%,1
20050 SOUND 2,VOL1%,A%,1
20060 SOUND 2,VOL1%,255,1
20070 SOUND 2,VOL1%,B%,1
20080 SOUND 2,VOL1%,A%,1
20090 SOUND 2,VOL1%,B%,1
20100 ENDPROC
20110  DEF PROCnoise2
20120 FOR A%=100 TO 200 STEP 25
20130 SOUND 1,VOL1%,A%-4,1:SOUND 1,VOL1
%,A%,1:SOUND 1,VOL1%,A%+4,1:SOUND 1,VOL
1%,A%+8,1
20140 SOUND 1,0,1,1
20150 NEXT
20160 ENDPROC
20170 ON ERROR OFF
20180 MODE7:IF ERR<>17 REPORT:PRINT" at
 line ";ERL
```



IF YOU WRITE TO US

BACK ISSUES (Members only)

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

NOTE OUR NEW SUBSCRIPTIONS ADDRESS

Subscriptions & Software Address
BEEBUG
PO BOX 109
High Wycombe
Bucks
HP11 2TD

SUBSCRIPTIONS

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

MEMBERSHIP COSTS:

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

SOFTWARE AND ROM OFFER (Members only)

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

IDEAS, HINTS & TIPS, PROGRAMS, AND LONGER ARTICLES

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

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

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

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Edited by Mike Williams. Technical Editor: Philip Le Grand.
Production Editor: Phyllida Vanstone. Technical Assistant: Alan Webster.
Managing Editor: Lee Calcraft.
Thanks are due to David Graham, Sheridan Williams, Adrian Calcraft, John Yale, Matthew Rapier 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) December 1983.

BEEBUG NEW ROM OFFER

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

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

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

Please note that we cannot accept EPROM-based operating systems (0.1 or 1.0) in lieu of payment. This can only be performed by Acorn dealers or by Acorn's service centre at Feltham, and applies only to users of the 0.1 O.S.

ADDRESS FOR 1.2 O.S. IF ORDERED ON ITS OWN:- ROM Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD. PLEASE ALLOW 28 DAYS FOR DELIVERY.

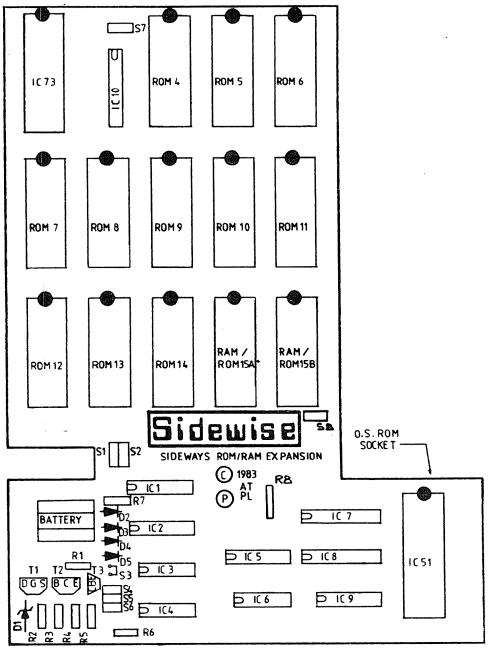
BEEBUGSOFT

ATPL'S SIDEWAYS ROM EXPANSION BOARD

SPECIAL PRICE TO MEMBERS £39.00 inc.

Save £5.70 on normal price of £44.70

* Simply plugs into the BBC Micro
* No soldering necessary
* Increases the sideways ROM capacity to 16
* Fully buffered - allows all sockets to be used
* Complete with full and detailed instruction booklet.
* Accepts 16K RAM in special sockets
* Battery back up facility for RAM
* As used at BEEBUG
* Reviewed in BEEBUG vol.2 number 6



HOW TO ORDER

Please send your order with a cheque / postal order made payable to BEEBUG, and enclose your membership number. We are unable to supply the board to overseas members.

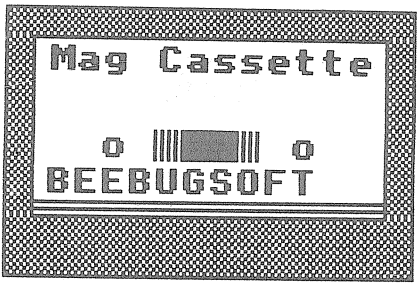
The address for SIDEWAYS is:
BEEBUGSOFT, PO Box 109, High Wycombe, Bucks. HP11 2TD.

MAGAZINE CASSETTE OFFER

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

Previous cassettes: Vol.1 No.10, Vol.2 No.1, Vol.2 No.2, Vol.2 No.3, Vol.2 No.4, Vol.2 No.5, Vol.2 No.6.

This month's cassette (Vol.2 No.7) includes: Computer Aided Design Program, Killer dice game, Teletext



(part 3) programs, Galactic Invasion game, Twelve Days of Christmas music, LINK utility for disc users, Percussion Machine, Screen Freezer, Chequer Board, Alien Destroyer Demo. All magazine cassettes cost £3.00 each. For ordering information see BEEBUGSOFT advertisement at the back of this month's magazine supplement.

MAGAZINE CASSETTE SUBSCRIPTION

We are able to offer members subscription to our magazine cassettes. Subscriptions will be for a period of one year and are for ten consecutive issues of the cassette. If required, subsriptions may be backdated as far as Vol.1 No.10, which was the first issue available on cassette. This offer is available to members only, so when applying for subscription please write to the address below, quoting your membership number and the issue from which you would like your subscription to start.

CASSETTE SUBSCRIPTION ADDRESS:

Please send a sterling cheque with order, together with your membership number and the date from which the subscription is to run, to:
PO Box 109, High Wycombe, Bucks, HP11 2TD.

CASSETTE SUBSCRIPTION PRICE:
UK £33 inc VAT and p&p
OVERSEAS (inc Eire) £39 inc p&p
(no VAT payable).

BEEBUG BINDER OFFER

BEEBUG MAGAZINE BINDER OFFER

A hard-backed binder for the BEEBUG magazine is now available. These binders are dark blue in colour with 'BEEBUG' in gold lettering on the spine. They allow you to use the whole of the first volume of the magazine as a single reference book. Individual issues may be easily added or removed. The binders will also conveniently hold less than 10 issues, so that you can use it while you build up Volume 2 also.

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

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

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