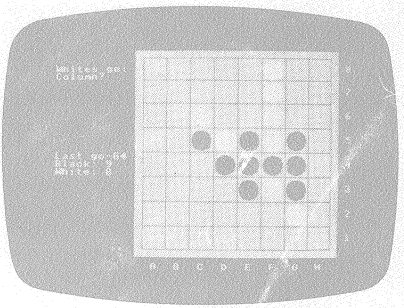
£1.00

BEEBUG FOR THE BBC MICRO

Vol 2 No 6 November 1983



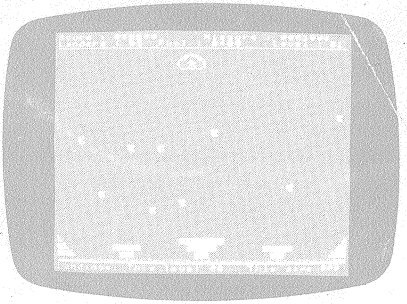
New Books Reviewed



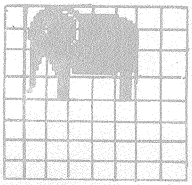
Reversi

JUMBO CHARACTER DEFINER





LUNAR ESCAPE



PLUS

- **DISC RECOVERY**
- **MODE 8**
- **ROM BOARDS REVIEWED**
- **INTERRUPT PROGRAMMING**
- **GAMES REVIEWED**
- **SLOW MOTION BEEB**
- **CLOCK DISPLAY**
- **TORCH BASIC REVIEWED**

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

EDITORIAL

You will see that BEEBUG has acquired a new editor. This does not imply any immediate or major changes nor does it signify any kind of palace revolution, but is just a consequence of the growth and success of BEEBUG and the launch of our new magazine for the Electron. I am also in no way related to Sheridan Williams, though our common surname does occasionally cause confusion in the office.

COMPUTER SHOWS

We would like to thank all those members who took the trouble to visit us on our stand at the PCW show. We shall also be at the 'Your Computer' Christmas Fair (see Events in the supplement) in December and hope to see as many as possible of you there. We shall be launching a range of software for Christmas at the show.

1.2 OPERATING SYSTEM ROM

For some time now, all BBC micros sold have come with O.S. 1.2 and many earlier models have been upgraded to take advantage of the extra features of the newer version. Since a very large majority of members must now be using O.S. 1.2 we have decided that from the start of 1984 we shall only use O.S. 1.2 to develop and test programs for the magazine. This enables us to use more of the features of the 1.2 operating system to the general benefit of programs, without the worry of incompatibility. In any case, more and more software suppliers will cease to cater for O.S. Ø.1, if they haven't already done so.

EXPANSION ROM BOARDS

Many BEEBUG members have expressed interest in expansion ROM boards for the Beeb and we have reviewed three in this issue. Acorn themselves do not make an expansion board for the BBC micro and have no immediate plans to do so.

However, it is quite likely that it will be difficult to fit both an expansion ROM board and one of the new double density disc controller boards (such as those from Microware and LVL) at the same time. We hope to be able to review double density disc controllers for the next issue of BEEBUG and will report then on any problems we find.

Mike Williams

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOAI

HINT WINNERS

This month's hint winners are G.M.Abernethy who wins the £1Ø prize, and Paul Beverley and J.L.Ward who both win a £5 prize.

MAGAZINE CASSETTE

This month the magazine cassette contains two extra items. The first is the winning program in our Sort Competition (see BEEBUG Vol.2 No.4) by J.B.Simpson. This is in assembler and is able to sort rapidly, data contained in numeric or string arrays. We are also including on the cassette full details on how to use the program. We have also included a demonstration of Swarmers' Revenge, BEEBUG's new game.

For those not wishing to purchase the whole tape, we are continuing our photocopy service (introduced last month) for the additional items on the magazine cassette (except for Swarmers' Revenge). Send 5Øp per item required, plus sae to: Photocopies, BEEBUG, PO Box 5Ø, St Albans, Herts. Please give your membership number.

BEEBUG POST

BEEBUG still requires a young technical assistant with software expertise on the Beeb. See the supplement for details.

NOTE - in this issue l and 1 are in some listings printed the same - there are no lower case Ls used on there own.

BEEBUG MAGAZINE

GENERAL CONTENTS

PAGE	CONTENTS
2	Editorial
4	News
5	Demolition
6	Mode 8 Screen Display
7	Multiple Character Definer
10	Nine Computer Games Reviewed
13	Mode 7 Clock Display
15	M-TEC BBC Basic(Z80) Review
16	An Introduction to Interrupt Programming
19	Writing a Game Program - Rapids
20	ROM Extension Boards Reviewed
23	Disc Snarfer
26	Three Books Reviewed
28	Using the Teletext Mode (Part 2)
31	The Beeb in Slow Motion
32	Postbag
33	Reversi
37	Lunar Escape
41	Points Arising

HINTS, TIPS & INFO

PAGE	CONTENTS
9	600 Baud RS423
12	Improved Error Handling
31	WORDWISE and Program Editing
36	FX Calls Greater than 166
36	Deleting Lines
36	Minute Print on an EPSON
41	The Four Ways of Using Function Keys
41	Double Quotes
41	TAB in WORDWISE

PROGRAMS

PAGE	CONTENTS
5	Demolition
6	Mode 8
7	Shaper
13	Clock Display
16	Interrupt Programs
19	Rapids
23	Snarfer
28	Teletext Programs
32	Slow Motion Beeb
33	Reversi
37	Lunar Escape

ADDITIONAL ITEMS ON MAGAZINE CASSETTE

- Fast Sort Routine
- Swarmers' Revenge Game Demo

NEWS

BEEBUG ON TV

Those of you who were able to watch the Thames Television programme "Database" on the evening of Wednesday 28th September will have seen copies of BEEBUG magazine being shown by Guy Kewney as examples of the best kind of magazine for the home and personal computer user. This occurred during an item on the recent PCW Show at the Barbican Centre, London at which BEEBUG had a stand. We noticed that many of you took time to visit us and we hope that your visit to the show was worthwhile.

BBC MICRO FOR THE US AND CHINA

The Wong Electronics Company of Hong Kong have won a US$45 million order to manufacture BBC micros for the US market. Shipping is scheduled to begin this autumn with 50,000 being produced in the next 12 months. Wong's already manufacture Acorn products for the Far East and Australia and negotiations are now in progress with China.

The US version of the micro will include disc drive, voice synthesis and Econet interface in the US$995 price. The machine will be aimed very strongly at the US educational market.

DISC EXECUTOR

Vision Software have just announced their software package called DISC EXECUTOR. It allows you to transfer any tape software to disc (including 'locked' programs), and allows games up to &6E blocks to be loaded and run from disc.

DISC EXECUTOR costs £11.00 from: Vision Software, 1 Allington Street, Liverpool, L17 7AD.

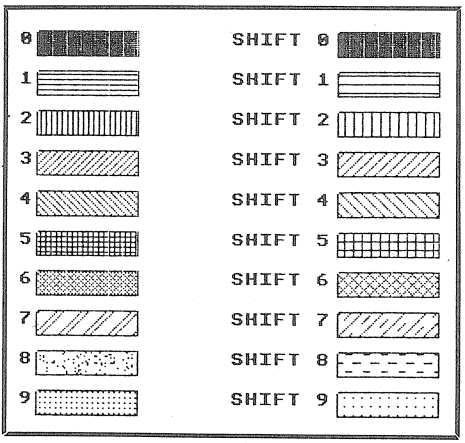
ACORN SHARES FOR SALE

Acorn Computers have recently announced that they are offering 10% of their shares for sale on the Unlisted Securities Market. So if you fancy a flutter in the stock market and a chance to get a stake in one of the top five computer munufacturers of this country, then see your stockbroker immediately.

COMPUTER AIDED DESIGN

Ibbotsons Design Software has recently released a CAD system, available on cassette, disc or EPROM called DIGITAL DRAWINGS. Some of the facilities offered are: 19 shading patterns, single key commands, reference scales and grids, 32 pre-defined graphics symbols, rubber banding and zoom. The package is designed to work with an EPSON FX-80 printer. The price will be below £100 for all the systems.

For further details contact Ibbotsons Design Software, The Byre, Ecclesbourne Lane, Idridgehay, Derbyshire, DE4 4JB.



MONITOR/TV STAND AND SOUND CONTROL

Microsupport are marketing a steel support for your monitor/TV. The BBC micro will slide underneath when not in use. It costs £16.99 inc. VAT and postage.

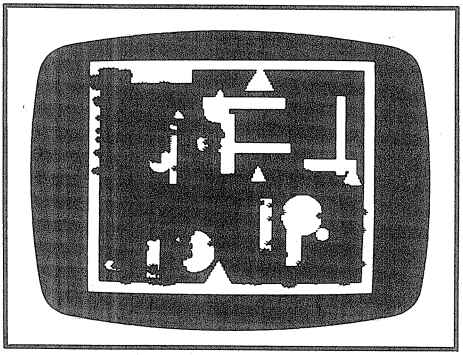
They also make an external volume control which allows you to connect your Beeb to another speaker, or from another socket, to an external amplifier. Cost is £4.00 inc VAT and post, from:
Microsupport, 104 Reddown Road, Coulsden, Surrey, CR3 1AL.

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

DEMOLITION (32k)

by D D Harriman

All at BEEBUG found this program highly entertaining when we first saw it in action. The colourful graphics, fast action and sizzling sound effects are all combined in a highly pyrotechnic display.



The program starts with a colourful display on the screen, consisting of randomly positioned shapes of various colours surrounded by a border. The computer then places somewhere on the screen, a small 'ball' which is then sent shooting across the display. Whenever it collides with a shape, or the border, an 'explosion' occurs, and part of the shape is destroyed. The ball bounces off and continues its destructive rampage around the screen, gradually eroding away every object in the display. Every collision is followed by a colour change on the screen. At times, everything seems to go berserk with a riot of flashing colours and explosive noises. The kaleidoscope of sound and colour continues for 5 minutes before a new screen is set up and the action starts all over again.

The sound effects are particularly effective, giving the impression of underground explosions. If you look carefully you will also see that the ball has a curved path ensuring, that it will never get caught in a repetitive sequence, unlike the old BREAKOUT game. This very modest program will well repay the small effort of typing it into your micro.

```
   10 REM Program DEMOLITION
   20 REM Version  B1.0
   30 REM Author D.D.Harriman
   40 REM BEEBUG November 1983
   50 REM Program subject to Copyright
   60 *FX9,12
   70 *FX10,12
   80 ENVELOPE 1,0,0,0,0,0,0,0,0,0,-1,-1,
-1,126,0
   90 MODE 2
  100 VDU 23,224,&5410;&BC7E;&7CFF;&104
E;
  110 VDU 23;8202;0;0;0;
  120 FOR Q%=1 TO 10
  130 H%=RND(300)+100
  140 V%=RND(300)+100
  150 IF H%<V% H%=30+RND(40) ELSE V%=30
+RND(40)
  160 X%=RND(1280-H%)-1
  170 Y%=RND(1024-V%)-1
  180 REPEAT N%=RND(15)
  190 UNTIL N%<>7
  200 GCOL 0,128+N%
  210 VDU 24,X%;Y%;X%+H%;Y%+V%;16,26
  220 GCOL 0,RND(6)
  230 X%=RND(1280)
  240 Y%=RND(1024)
  250 Z%=RND(100)
  260 MOVE X%,Y%
  270 MOVE X%+2*Z%,Y%
  280 PLOT 85,X%+Z%,Y%+2*Z%
  290 GCOL 0,RND(6)
  300 X%=RND(1280)
  310 Y%=RND(1024)
  320 Z%=RND(100)
  330 FOR F%=0 TO Z% STEP 4
  340 K%=SQR(Z%*Z%-F%*F%)
  350 MOVE X%-K%,Y%+F%
  360 DRAW X%+K%,Y%+F%
  370 MOVE X%-K%,Y%-F%
  380 DRAW X%+K%,Y%-F%
  390 NEXT
  400 NEXT
  410 GCOL 0,132
  420 VDU 24,0;0;50;1023;16
  430 VDU 24,0;0;1270;50;16
  440 VDU 24,1230;0;1279;1023;16
  450 VDU 24,0;974;1279;1023;16
  460 VDU 20,26
```

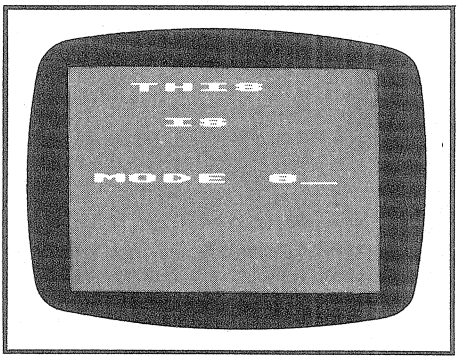
```
  470 X%=100
  480 Y%=900
  490 X=8
  500 Y=1
  510 TIME=0
  520 REPEAT Y=Y*.999-.1
  530 X=X*.999
  540 P%=POINT(X%+X,Y%)
  550 IF P%<>7 AND P%<>0 X=-X:PROCZ ELS
E P%=POINT(X%,Y%+Y):IF P%<>7 AND P%<>0
Y=-Y:PROCZ
  560 PLOT 71,X%,Y%
  570 X%=X%+X
  580 Y%=Y%+Y
  590 PLOT 69,X%,Y%
  600 UNTIL TIME>30000:RUN
  610 DEF PROCZ
  620 VDU 19,P%,RND(6);0;
  630 X=X*1.03
  640 Y=Y*(1.03-.1*(Y>0 AND Y%<500))
  650 MOVE X%-32+RND MOD 9,Y%+12+RND MO
D 5
  660 GCOL 0,0
  670 VDU 5,224,4
  680 GCOL 0,7
  690 SOUND 16,1,3+RND(3),1
  700 IF X%<50X=8 ELSE IF X%>1230 X=-8
  710 IF Y%<50 Y=8 ELSE IF Y%>974 Y=-8
  720 IF ABS X>16 X=16*SGN(X)
  730 IF ABS Y>16 Y=16*SGN(Y)
  740 ENDPROC
```

This will work on O.S. 1·2 and Basics I and II

MODE 8 SCREEN DISPLAY (16k)

As an example of the detailed information contained in the Advanced User Guide (reviewed elsewhere in this issue), the publishers have given us permission to print this extract in BEEBUG.

This example program will set up a brand new mode which has been nominated as 'MODE 8'. It is acheived by directly accessing the video ULA. 'MODE 8' is a full 16 colour mode and can be implemented on a Model A as well as a Model B since only 10k of RAM is used. There will only be 10 characters across the screen, but printing and plotting will operate properly. One word of warning however. Do not try to redefine the text window when this mode is in use because it will not work! All error checking on window bounds will be ineffective in this mode since the operating system does not expect mode 8 to exist.




```
  10 REM CREATE 'MODE 8'-
  20 REM  NOTE: NO WINDOWING ALLOWED
  30 REM
  40 REM  NOTE: POKING VDU VARIABLES
  50 REM   IS GENERALLY ILL ADVISED.
  60 MODE 5:REM BASIC MODE
  70 REM CONFIGURE VIDEO ULA FOR 10 CO
LUMN, 16 CHARACTER
  80 *FX 154,224
  90 ?&360=&F:REM COLOUR MASK
 100 ?&361=1:REM PIXELS PER BYTE-1
 110 ?&34F=&20:REM BYTES PER CHARACTER
(4 WIDE x 8 HIGH)
 120 ?&363=&55:REM GRAPHICS RIGHT MASK
 130 ?&362=&AA:REM GRAPHICS LEFT MASK
 140 ?&30A=9:REM No. OF CHARS PER LINE
 150 VDU 20
 160 REM DEMO
 170 MOVE 0,0:DRAW 640,512:DRAW 1279,0
 180 PRINT TAB(1,2);
 190 A$="***HelloThere***"
 200 COLOUR 129
 210 FOR A%=1 TO 16
 220 IF A%=9 THEN PRINT TAB(1,8);
 230 COLOUR A%-1
 240 PRINT MID$(A$,A%,1);
 250 NEXT A%
 260 PRINT
```

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

MULTIPLE CHARACTER DEFINER (32k)

by Ian McEwen

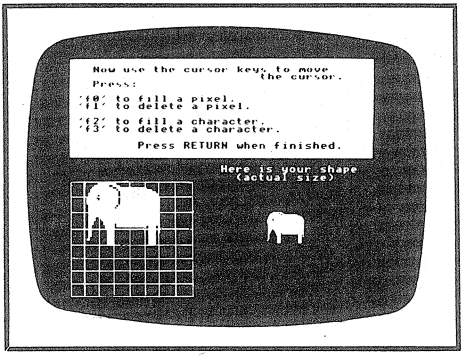
The ability to define your own characters is one of the most useful features of the Beeb. Many games programs make considerable use of this facility. Here we present a program which not only makes this task easy but enables you to build larger shapes using multiple user defined characters.

SHAPER is a useful utility program that allows you to build up multiple user defined characters that you can then incorporate into other programs. You are presented with a grid of 64 empty characters on the screen which you can then use to design your own shapes, moving about by means of the cursor keys. As you define the characters on this grid, the same characters, actual size, are displayed at the right hand side of the screen. A significant advantage of this utility is the ability to define a large shape comprised of several individual characters as a whole, instead of having to define each character individually.

When you have finished creating the shape of your choice, the parameters for each defined character are listed on the screen and, optionally, on a printer. The eight parameters listed for each character can then be used to define the same character in another program using the VDU23 command (see pages 170 to 179 and page 384 of the User Guide for further information).

To use the SHAPER program, you will need first to type it in and save it on tape or disc. Disc users will need to set PAGE lower than the default &1900 (&1200 is recommended) in order to use the program. Ideally, a move down routine should be used which avoids any problems of program corruption by the DFS. Otherwise, load and run the SHAPER as for any Basic program.

Detailed instructions for using SHAPER are contained in the screen displays. When using the cursor keys, keep any key continually pressed to move quickly across the grid. In any cursor position, pressing 'f0' will display a single pixel in that position, and pressing 'f1' will delete a pixel. Trying to insert a pixel where



one already exists or to delete a non-existant pixel will produce a warning bleep. The function keys f2 and f3 can be used to display, or delete, all the 64 pixels comprising a complete character. This feature can also be used to define black shapes on a white background. Experimenting with the cursor and function keys will soon demonstrate the power and convenience of this program. Using SHAPER, any shape can be rapidly drawn as a set of user defined characters.

Pressing Return will terminate character editing and the corresponding VDU 23 codes for each defined character are then listed on the screen, and copied to a printer if required. The codes for each character can then be used as the parameters of VDU 23 commands to define the same characters in other programs. Normally ASCII codes in the range 224 to 255 are used for this purpose (see pages 170 to 179 and page 384 of the User Guide for further information). You will also need to note the relative positions of several characters that are to be combined together in order to preserve this relationship in future use. This information is also displayed on the screen along with the character definitions.

NOTES FOR MORE ADVANCED USERS.

Although the SHAPER program allows up to 64 new characters to be defined at any one time you will normally be limited to using only 32 characters with ASCII codes 224 to 255. However, under version 1.Ø and later of the operating system, you can define more than 32 characters.

In addition to ASCII codes 224 to 255 it is also possible to redefine the characters with ASCII codes 128 to 159 while, if additional memory is reserved for the purpose, all ASCII codes in the range 32 to 255 can be redefined. For further information see pages 427 and 428 of the User Guide.

```
  10 REM Program   SHAPER
  20 REM Version   B1.9
  30 REM Author    I.McEwan
  40 REM BEEBUG    November 1983
  50 REM Program Subject to Copyright
  60 :
 100 MODE7
 110 ON ERROR GOTO 1740
 120 *FX15,1
 130 PROCfrontpage
 140 :
 150 MODE1
 160 PROCinit:PROCtextwindow(Ø,12,39,Ø)
 170 PROCgraphwindow(Ø,Ø,1279,576)
 180 PROCframe(Ø,Ø,ht,(wt-ht),nh,nv)
 190 PROCdraw
 200 *FX15,1
 210 *FX4
 220 *FX12,Ø
 230 PRINT''"Press any key to exit.":X
=GET
 240 MODE7
 250 END
 260 :
1000 DEFPROCinit
1010 *FX225,128
1020 sq=224:cur=225:@%=Ø:nh=8:nv=8:ht=
512:wt=512:Pflag%=FALSE
1030 VDU23,sq,192,192,Ø,Ø,Ø,Ø,Ø,Ø
1040 VDU23,cur,Ø,192,Ø,Ø,Ø,Ø,Ø,Ø
1050 VDU19,2,7,Ø,Ø,Ø
1060 DIMpos%(64),defchr%(64,8):ENDPROC
1070 :
1080 DEFPROCgraphwindow(a,b,c,d)
1090 VDU24,a;b;c;d;:ENDPROC
1100 :
1110 DEFPROCtextwindow(a,b,c,d)
1120 VDU28,a,b,c,d:ENDPROC
1130 :
1140 DEFPROCframe(x,y,s,r,h,v)
1150 LOCAL K%
1160 VDU5:GCOL3,1:MOVEx,y
1170 DRAW x,y+s:DRAW x+s+r,y+s:DRAW x+
s+r,y:DRAW x,y
1180 FORK%=1TO(v-1):MOVE x,y+(s/v)*K%:
DRAW x+s+r,y+(s/v)*K%:NEXT
1190 FOR K%=1TO(h-1):MOVE x+((s+r)/h)*
K%,y:DRAW x+((s+r)/h)*K%,y+s:NEXT
1200 GCOLØ,3:VDU4:ENDPROC
1210 :
1220 DEFPROCdraw
1230 LOCAL H,V,m%,n%
1240 COLOUR 131:COLOUR Ø:CLS
1250 PRINT'TAB(3);"Now use the cursor
keys to move"
1260 PRINTTAB(25);"the cursor."'TAB(3)
;"Press:-"
1270 PRINT'" 'fØ' to fill a pixel."
1280 PRINT" 'f1' to delete a pixel."
1290 PRINT'" 'f2' to fill a character.
"
1300 PRINT" 'f3' to delete a character
."
1310 PRINT'TAB(9);"Press RETURN when f
inished."
1320 *FX4,1
1330 *FX12,2
1340 VDU5 :GCOLØ,3:MOVE 64Ø,576:PRINT"
Here is your shape":MOVE 704,544:PRINT"
(actual size)"
1350 H1=32:V1=480
1360 H=H1-32:V=V1+32:h=1:v=1:GCOL 3,2
1370 REPEAT
1380 *FX15,1
1390 MOVE H,V:VDUcur:m%=GET:MOVE H,V:V
DUcur
1400 IF m%= 136 THENh= h-1:H=H-8:IFh<1
THEN h=8:H1=H1-64:IF H1<Ø THEN h=1:H=H+
8:H1=H1+64
1410 IF m%=137 THENh=h+1:H=H+8:IFh>8TH
EN h=1:H1=H1+64:IF H1>wt THEN h=8:H=H-8
:H1=H1-64
1420 IF m%=138 THEN v=v+1:V=V-8:IF v>8
 THEN v=1:V1=V1-64:IF V1<Ø THEN v=8:V=V
+8:V1=V1+64
1430 IF m%= 139 THEN v=v-1:V=V+8:IF v<
1 THEN v=8:V1=V1+64:IF V1>ht THEN v=1:V
=V-8:V1=V1-64
1440 h%=(H1+32)/64:v%=(V1+32)/64
1450 n%=h%+(8-v%)*8
1460 IF m%=128 THEN PROCcheck(3,h,v):I
F cancel%=1 THEN GCOL3,2:GOTO 1520
1470 IF m%=128 THEN PROCplace
1480 IF m%=129 THEN PROCcheck(Ø,h,v):I
F cancel%=1 THEN GCOL3,2:GOTO 1520
1490 IF m%=129 THEN PROCdelete
1500 IF m%=130 THEN PROCchange(3)
1510 IF m%=131 THEN PROCchange(Ø)
1520 UNTIL m%=13
1530 PROCchr:ENDPROC
1540 :
```

```
1550 DEFPROCchr
1560 LOCAL K%,X,X$
1570 GCOL0,3 :VDU4:PROCtextwindow(0,12
,39,0)
1580 CLS:PRINTTAB(3,7);"Press 'P' for
a print-out"
1590 PRINT'TAB(3);"else the SPACE BAR.
";:X$=GET$
1600 IF X$="P" OR X$="p" THEN CLS:PRIN
T'''''TAB(5);"Switch your printer on.":
PRINT''''TAB(12);"Press Space Bar when
ready.":X=GET:Pflag%=TRUE
1610 PROCtextwindow(0,22,39,0)
1620 COLOUR 128:COLOUR 3
1630 CLG:CLS:PROCframe(0,0,256,256,nh,
nv):VDU14:IF Pflag% THEN VDU2
1640 PRINTTAB(5);"Here are your defini
tions:-"
1650 FORK%=1 TO nh*nv
1660 IF pos%(K%)>0 THEN PROCdefprint
1670 NEXT K%
1680 PRINT:VDU5,3:K%=1:REPEAT
1690 IF pos%(K%)>0 THENMOVE((K%-1)MOD8
)*64,254-((K%-1)DIV8)*32:PRINTK%
1700 K%=K%+1:UNTIL K%>nh*nv
1710 MOVE 550,200:PRINT"The arrangemen
t of":MOVE 550,136:PRINT"your character
s is":MOVE 550,76:PRINT"shown in this f
rame.":VDU4
1720 ENDPROC
1730 :
1740 ON ERROR OFF
1750 MODE7
1760 VDU26,4,30
1770 *FX4
1780 *FX12,0
1790 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
1800 END
1810 :
1820 DEFPROCcheck(clr%,h,v)
1830 LOCAL h%,v%:cancel%=0
1840 h%=787+4*h+H1/2:v%=160-4*v+V1/2:I
F POINT(h%,v%)=clr% THEN cancel%=1:IF m
%=129 THEN VDU7
1850 ENDPROC
1860 :
1870 DEFPROCfrontpage
1880 LOCAL K%
1890 FOR K%=1 TO 2
1900 PRINTTAB(5,K%);CHR$132CHR$157CHR$
131CHR$141;TAB(10);"S H A P E R";TAB(30
);CHR$156
1910 NEXT
1920 PRINT''TAB(10);"This program help
s in"
1930 PRINT'TAB(5)"the composition of S
HAPES using"
1940 PRINT'TAB(4)"multiple user-define
d characters."
1950 PRINT'TAB(7);"(User Guide pages 3
84-385)"
1960 PRINT'''
1970 PRINTSPC(8);CHR$157;CHR$135;CHR$1
32;"Press the SPACE BAR";SPC(2);CHR$156
:X=GET
1980 ENDPROC
1990 :
2000 DEFPROCdefprint
2010 LOCAL J%,K$
2020 K$=STR$(K%):IF K%<10 THEN K$=CHR$
32+K$
2030 PRINT'"No."+K$+":- ";
2040 FORJ%=1TO8:PRINT defchr%(K%,J%);
2050 IF J%<8 THEN PRINT",";
2060 NEXTJ%
2070 ENDPROC
2080 :
2090 DEF PROCplace
2100 pos%(n%)=pos%(n%)+2^v:defchr%(n%,
v)=defchr%(n%,v)+2^(8-h):P%=1:MOVE H,V:
GCOL3,3:VDUsq:GCOL0,3:PLOT69,787+4*h+H1
/2,160-4*v+V1/2:GCOL 3,2
2110 ENDPROC
2120 :
2130 DEF PROCdelete
2140 pos%(n%)=pos%(n%)-2^v:defchr%(n%,
v)=defchr%(n%,v)-2^(8-h):D%=1:MOVE H,V:
GCOL3,3:VDU sq:PLOT69,787+4*h+H1/2,160-
4*v+V1/2:GCOL3,2
2150 ENDPROC
2160 :
2170 DEF PROCchange(X)
2180 LOCAL h,v,H,V
2190 FOR h=1 TO 8:FORv=1 TO 8
2200 PROCcheck(X,h,v)
2210 H=H1+8*(h-1)-32:V=V1-8*(v-1)+32
2220 IF cancel%=0 THEN PROCcng(X)
2230 NEXT v,h
2240 ENDPROC
2250 DEF PROCcng(X)
2260 IF X=3 THEN PROCplace ELSE PROCde
lete
2270 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

600 BAUD RS423 - Nick Porter

The Tandy CGP-115 runs at 600 baud. To run the Beeb's RS423 interface at 600 baud requires a simple trick. Set up the baud rate at 150 baud and then alter the clock division ratio in the 6850 ACIA chip with:

```
*FX156,1,253
```

NINE COMPUTER GAMES REVIEWED

BY Robert Barnes & Alan Webster

Program : Bandits at 3 O'Clock
Supplier: Program Power
Price : £6.95
Rating : ***

Bandits at 3 o'clock is a new game for one or two players in which you control a bi-plane. You will need joysticks, as this is the only form of input allowed by this program, but controlling a plane is much easier this way than using the standard keyboard.

The display consists of a church and grave yard at the bottom centre of the screen, and two runways at the extreme edges of the screen. The idea of the game is to try to destroy your enemy using a machine gun mounted in your plane.

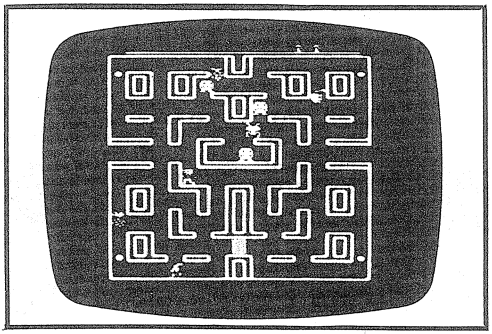
The game has many options including choice of day or night and additional problems in the form of clouds, flak and airships. This is a simple but entertaining game, which makes a change from the pressures of the normal arcade games.

Program : Attack on Alpha Centauri
Supplier: Software Invasion
Price : £7.95
Rating : ****

Attack on Alpha Centauri is a new game from Software Invasion, who specialise in producing high quality three dimensional graphical games. The game features some of the best and most unusual graphics yet for the BBC micro.

This is a version of Galaxians the like of which you've never seen before. An alien landscape is displayed which has been very well designed using the technique of overlaying colours to produce many different shades. When the game starts, one of several volcanos errupts and impressive looking insects fly out which you then have to shoot down. As the insects come for you they actualy increase in size until they disappear over the horizon. Some of these effects are really very impressive.

This game, like the others so far produced by Software Invasion, is an excellent example of what can be achieved on the BBC and is highly recommended.



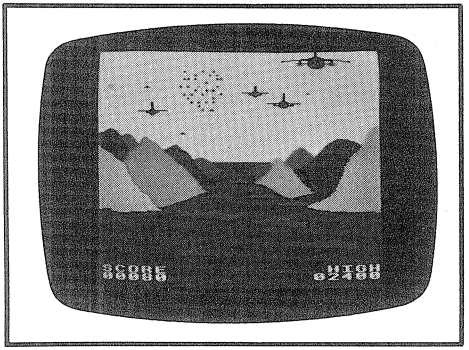
FELIX AND THE FRUIT MONSTERS

Program : Felix and the Fruit Monsters
Supplier: Program Power
Price : £7.95
Rating : **

Felix and the fruit monsters is yet another program based on the Snapper type of maze game. This one does have a difference in that you have to protect a fruit from marauding monsters until the on-screen timer reaches zero.

In my view, this game lacks the originality to make it as worthwhile as some of the other programs on the market, even though the game does get slightly better as it progresses through its different stages. The sound soon becomes quite irritating but fortunately this can be turned off. This is not, as far as I am concerned, one of Program Power's better games, and does not come close to chef d'oeuvres like Killer Gorilla.

Program : 3D Bomb Alley
Supplier: Software Invasion
Price : £7.95
Rating : ****



3D Bomb Alley is another 3 dimensional program from Software Invasion, with graphics almost as good as those of Attack on Alpha Centauri.

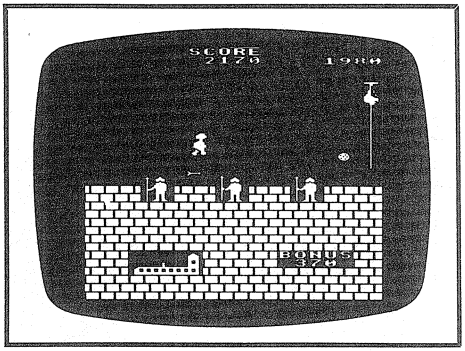
Your task in this game is to protect a fleet of ships from attacks by a number of relentless fighter aircraft. You control a battery of ground to air missiles, using either joysticks or the keyboard. This might sound easy, but wait until there are four or five bombers coming at you at once. This is a very addictive game and is good value for money.

Program : Painter
Supplier: A and F Software
Price : £8.00
Rating : ***

In this game you are a painter whose job is to fill in boxes by completing the lines surrounding them. At the same time you are being hunted by 'chasers' which you must avoid. Your only protection, other than speed and cunning, is the ability to insert 3 gaps in the lines, which the Chasers cannot cross. When you complete the lines the boxes change colour, and slowly a character is built up. For example a space invader character and the A & F logo are formed.

I didn't find this game very fast even though it is written in machine code and it doesn't stand out in this highly competitive market.

Program : Hunchback
Supplier: Superior Software
Price : £7.95
Rating : ****



Hunchback is one of two new releases from Superior Software. It is an extremely good version of the arcade game where Quasimodo has to cross a series of walls in order to rescue his beloved Esmerelda. There are bonus points for completing each wall as quickly as possible, but you have to concentrate on the game so much that you forget about the bonus.

There are 12 walls to navigate, and once you have done this, you go back to the first wall and the game gets quicker (much quicker!).

This game makes a nice change from the "Blast-'em-out-of-the-sky" type games and is thoroughly recommended.

Program : Felix in the Factory
Supplier: Program Power
Price : £7.95
Rating : ****

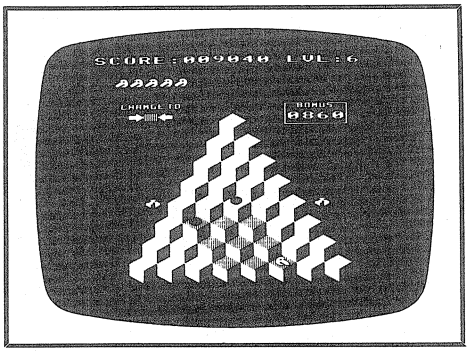
Felix in the factory is one of the new games on the market which takes its origins from the Monsters / Krazy Kong type games which involve ladders and walkways.

The idea for this game is very novel and the graphics are good. You take the part of Felix who has to run around the conveyor belts, up the ladders and along the walkways to get a can of oil with which he can re-fuel the generator. As time goes by, the generator slowly runs out of oil, and once run out, Felix dies.

Rats, and monsters of all kind

hamper you in your quest to keep the generator going. All in all, this is a very good game, but the key layout left me with a sore hand.

Program : Q*bert
Supplier: Superior Software
Price : £7.95
Rating : ****



Q*bert is the other new release from Superior Software and it is as every bit as good as Hunchback.

The game is difficult to explain, but in principle you (with a very long nose indeed) move over a 3D pyramid, changing the colour of the slab that you are on. Once all of the slabs have changed colour, you proceed to the next level. Sounds easy? Well there are a few snakes which are out to get you. If this still sounds easy then buy the game and find out!

As with all Superior Software games, you can choose the starting level and there is also a high score table. This is very good value for money indeed.

Program : Bug Blaster
Supplier: Alligata
Price : £7.95
Value : *****

Bug Blaster is one of the best games around for the BBC micro. It is fast, colourful and fun to play. It is one of the most addictive games I have ever played (and the most addictive one that our editor has played!).

It is a very good version of the arcade game 'Centipiede' in which you are on patrol in a field full of mushrooms. Your aim is to avoid all of the creatures that roam around, and score as many points as possible.

There is a centipede which gradually makes it way down the screen, a spider which moves across the bottom of the screen, fleas which fall from top to bottom and a scorpion, which if it touches a mushroom poisons it. If a centipede hits a poisoned mushroom it heads straight down the screen towards you. This is another program which is excellent value for money.

SPECIAL OFFER

We have again arranged special offers for BEEBUG members on the best of the software reviewed in this issue. Details of this month's offer will be found in the magazine supplement under 'Software Club'.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

IMPROVED ERROR HANDLING - G.M.Abernethy

The routine given does the usual printing of the error message and line number but follows this by listing the line where the error occurred automatically. This is achieved through the clever use of *FX138. To use it insert:

```
10 ON ERROR PROCerrorhandler:END
```

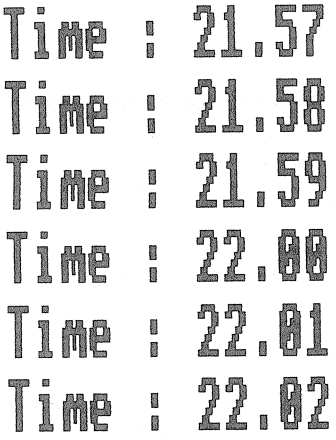
with the error handling procedure at the end of the program

```
1000DEF PROCerrorhandler
1010ON ERROR OFF
1020REPORT
1030PRINT " at line ";ERL
1040*FX138,0,76
1050*FX138,0,73
1060*FX138,0,83
1070*FX138,0,84
1080A$=STR$(ERL)
1090FOR K%=1 TO LEN(A$)
1100A%=138
1110X%=0
1120Y%=ASC(MID$(A$,K%,1))
1130CALL&FFF4
1140NEXT K%
1150*FX138,0,13
1160ENDPROC
```

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

MODE 7 CLOCK DISPLAY (16k)

by D P Wright



This program displays a continuously updated digital clock at the top of the screen in mode 7. However, it is quite invisible to Basic, and you can run Basic programs in mode 7, or perform program editing etc without affecting the clock display.

If you would like to have a digital clock displayed on the screen, just type in the program and save it on tape or disc. Then try running it and see what happens. With machine code programs it is always worth saving the first version before trying it out, just in case you have made a typing mistake. To save time, you can omit all lines where \ is the first character after the line number, as these are just comments. When the program is run it will prompt for the current time in hours, minutes and then seconds. Enter these in standard 24 hour clock format. When Return is pressed after entering the seconds, the clock is activated. This results in the screen clearing and the time being displayed in white on blue in a stripe at the top of the screen in mode 7. This is updated once every second.

Once the program is working (when all typing errors have been cured), it is possible just to save the piece of machine code that displays the time. To do this you should first save the original program listing for safety, and then run it to assemble the code into memory. When this has finished press Escape in response to the 'hours' prompt and type:

```
*SAVE TIMER CØØ +DF
```

This will save the code on its own. To use the clock at some other time all you need to do is *LOAD the program into memory. Set the variables A%, X% and Y% to the hours, minutes and seconds respectively and then type:

```
CALL &CØØ
```

to set the timer. Having initiated the timer, you can ignore its operation and other programs can be run independently. The only rules you must follow are not to change mode and not to use the user defined graphics characters as these are stored in the memory space where the machine code is executing. Since these are not effective in mode 7 there should be no conflict here. Note too, that you only have 24 lines to use on the screen and not 25, as the top line is now reserved for displaying the clock.

When the timer is running, operation of the cassette and disc systems etc. are unaffected. However, should you wish to disable the clock, simply type *FX13,5. It can be restarted, as described before, with CALL&CØØ, after setting A%, X% and Y% to the hours, minutes and seconds as required.

For the more technically minded, we provide an explanation of the operation of the program.

Essentially, the program sets up the Beeb's interval timer to generate an event every second. This event is then used to display the time. The actual time display is done through direct pokes into the video display as OSWRCH is a non re-entrant routine and thus gets confused if this call is used whilst the foreground process (your program) is printing a character as well. This method can lead to other problems if trying to display characters in a scrolling area of the screen. In this program the text window is redefined so that the top line does not scroll.

When the program is called (START at line 860) it jumps to the initialisation routine. This sets up the time stores and then selects mode 7 with the appropriate text window to protect the display. The event vector is then set up at &220 before initialising the interval timer and enabling the event.

In operation the time is displayed on the screen and the interval timer is set going. One second later the interval timer crosses zero and an event is generated. When this occurs control is passed to the interrupt routine (ROUTINE at line 170) pointed to by the vector. The interval timer is set back to 1 second before zero. The clock value is then incremented in its internal version, with checking for the correct bounds for minutes and seconds. These values are then converted to BCD and then ASCII before being poked straight into the screen memory.

The generation of an event is of much more general application. With this it would be possible to poll devices like thermometers, pumps and so on, once every second to run the central heating, and this would again be performed completely invisibly.

```
 10  REM Program TIME
 20  REM Version B1.0
 30  REM Author D.P.Wright
 40  REM BEEBUG November 1983
 50  REM Program subject to copyright
 60  :
 70 REM Disable event for re-assembly
 80 *FX13,5
 90 OSBYTE=&FFF4
100 OSWRCH=&FFEE
110 OSWORD=&FFF1
120 FORI%=0TO1
130 P%=&C00
140 [OPTI%*3
150 JMP START
160 :
170 .ROUTINE
180 \--clock int. service routine--
190 \--Save contents of all regs.--
200 PHP:PHA
210 TXA:PHA:TYA:PHA
220 \---reset clock---
230 LDX #CLOCK MOD 256
240 LDY #CLOCK DIV 256
250 LDA #4
260 JSR OSWORD
270 \----increment time----
280 INC SECOND
290 LDA SECOND:CMP #60:BNE a
300 LDA #00:STA SECOND
310 INC MINUTE
320 LDA MINUTE:CMP #60:BNE a
330 LDA #00:STA MINUTE
340 INC HOUR
350 LDA HOUR:CMP #24:BNE a
360 LDA #00:STA HOUR
370 :
380 .a
390 \----display clock on screen----
400 \--Set back/foreground colours--
410 LDA #132:STA HIMEM+40
420 LDA #157:STA HIMEM+41
430 LDA #135:STA HIMEM+42
440 LDY#57
450 LDA HOUR:JSR OUTPUT
460 LDA #58:STA HIMEM,Y:INY
470 LDA MINUTE:JSR OUTPUT
480 LDA #58:STA HIMEM,Y:INY
490 LDA SECOND:JSR OUTPUT
500 \---restore contents of regs.---
510 PLA:TAY:PLA:TAX:PLA:PLP
520 RTS
530 :
540 \----splits value in Acc.----
550 \----into 2 BCD numbers  ----
560 \----and outputs ASCII   ----
570 \----values on screen.   ----
580 .OUTPUT
590 LDX #0
600 SEC
610 .loop SBC #10
620 BMI b
630 INX:JMP loop
640 .b DEX
650 CLC
660 ADC #58
670 PHA
680 TXA:ADC #48
690 STA HIMEM,Y: \--MSB--
700 INY:PLA
710 STA HIMEM,Y: \--LSB--
720 INY:RTS
730 ]
740 :
750 REM---data for clock and time---
760 CLOCK=P%
770 !CLOCK=&FFFFFF9C
780 ?(CLOCK+4)=&FF
790 P%=P%+5
800 HOUR=P%
810 MINUTE=P%+1
820 SECOND=P%+2
830 P%=P%+3
840 [OPTI%*3
850 :
```

```
  860 .START
  870 \---initialisation routine---
  880 \---Initialise time---
  890 STA HOUR:STX MINUTE:STY SECOND
  900 \---Mode7---
  910 LDA #22:JSR OSWRCH
  920 LDA #7:JSR OSWRCH
  930 \---Change text window---
  940 LDA #28:JSR OSWRCH
  950 LDA #0:JSR OSWRCH
  960 LDA #24:JSR OSWRCH
  970 LDA #39:JSR OSWRCH
  980 LDA #2:JSR OSWRCH
  990 :
 1000 \--Set add. of int.service rout.-
-
 1010 LDA #ROUTINE MOD 256
 1020 STA &220
 1030 LDA #ROUTINE DIV 256
 1040 STA &221
 1050 JSR ROUTINE
 1060 \---Enable clock interrupt---
 1070 LDA #14
 1080 LDX #5
 1090 JSR OSBYTE
 1100 RTS
 1110 ]
 1120 NEXT
 1130 REM Set up clock and start it
 1140 INPUT'"HOURS   ",A%
 1150 INPUT"MINUTES ",X%
 1160 INPUT"SECONDS ",Y%
 1170 CALL&C00
```

M – TEC BBC BASIC (Z80) REVIEW

by Colin Opie

In BEEBUG Vol.2 No.1 we reviewed the Torch Z80 Disc Pack. A complete Z80 version of BBC Basic is now available for the Torch system and your Torch second processor can now have BBC Basic together with a full use of Sound, Graphics, and in-line Z80 assembler, and this combination provides you with at least 47k of user memory.

The new version is indeed a complete BBC Basic. It is difficult to say whether it is Basic-I or BASIC-II because, for example, it has the new OSCLI statement but not OPENUP! This aside, the upgrade is extremely good. In the few cases where statement syntax is not completely identical, it is usually because the Z80 version is providing a more relaxed or versatile format.

BBCBASIC(Z80) comes with a number of utilities. The most useful of these are a CONVERT program for converting to internal format Basic files from ASCII files and vice versa, a RUN utility for getting back into your program after a break to CPN (CPN is the Torch disc operating system), an UNLIST utility for creating unlistable programs, a MERGE program for merging Basic programs, and a series of HELP text files. A well-written and thorough manual accompanies the software.

FIRST THOUGHTS

To get a feeling for how good the system was I transferred a number of Graphics and Sound programs from Acorn DFS discs onto a Torch CPN disc using the TORCH RDACORN utility. I then used the CONVERT utility to obtain the equivalent internal format Basic files. [Hint: The CONVERT utility will create lines for the 'LIST' and '*SPOOL' commands used initially to create and close off the spooled file used by RDACORN. It is therefore a good idea to start your programs from some line number greater than one]. I then loaded BBCBASIC(Z80) and, after deleting the two redundant lines, proceeded to run each of the Basic programs - with no problems whatsoever!

COMPARISONS

In BBCBASIC(Z80) the DELETE, RENUMBER, and LIST statements will accept either a comma ',' or a hyphen '-' between arguments. This is good news for those of us who have used a number of other Basics. LISTO7 actually indents the FOR..NEXT and REPEAT..UNTIL loops correctly, so that the two statements come into line.

The built-in Z80 assembler uses the same format for creating in-line assembly code (eg. in the way square

Contd on p. 25

AN INTRODUCTION TO INTERRUPT PROGRAMMING (16k)

by Trevor Pullen

The last example program accompanying this article, called INTRP3, is interesting in its own right. If you type it in and run it you will then find you have a Beeb with a musical keyboard!

This will work on O.S. 1.2 Basics I and II

Interrupt programming is a subject which is often avoided, and is treated by many with much uncertainty - an unfortunate state of affairs since the use of interrupts is fundamental to many forms of interactive and real-time computing applications. In this article, which introduces the subject, Trevor Pullen explains what interrupts are, how they may be generated and shows you how to write simple programs using them. Although the interrupt handling routines have to be in machine code, they are quite short and can easily be incorporated into any Basic program as we show here.

WHAT ARE INTERRUPTS?

As their name suggests, their purpose is to interrupt or alert the processor to a particular event having occurred. The event causing the interrupt may be either internally or externally generated and need have no bearing or relation to the current status of the processor. The important point is that on receipt of the interrupt, the processor will leave its current task, service the interrupt, and then resume computation where it last left off. Hardware interrupts on the Beeb are of two kinds called IRQ and NMI. These are very similar except that NMIs (Non Maskable Interrupts) cannot be ignored whereas IRQs (Interrupt Requests) can be disabled by software. The BBC micro also provides a 'packaged' set of interrupts called events, which are much easier to use.

A computer that uses interrupts will often give the appearance of being able to carry out several tasks simultaneously. The BBC micro, for example, can still accept input from the keyboard even when it is busy doing something else, like listing a program on the screen. The information you type in is not lost, but carefully stored away for use later on. The example programs in this article will illustrate this point even more effectively. Hopefully, even from this brief introduction, the potential offered by the interrupt facility can readily be seen.

Before we consider any programs in detail, there are a number of general points that need to be made.

(a) There are limits to the time that can elapse between the generation of an interrupt and the return of control to the main user program. In the BBC User Guide, page 465, a maximum time of less than 1 millisecond is recommended. It should be remembered though, that since the 6502 processor runs at 2MHz, this gives time for at least 600 instructions to be executed, which should be sufficient for most applications!

(b) On entering the interrupt handling routine, the current values of the A, X and Y registers, together with the processor status byte, must be saved, to be restored to their original values once the interrupt has been serviced.

(c) It is a characteristic of the BBC machine that 'interrupts' and 'events' indirect via locations in page 2 of memory. It is thus necessary to make the relevant location (vector) point to the beginning of the user routine. Further, before enabling any events, the computer must be set in the binary mode (and not BCD - Binary Coded Decimal - mode).

(d) At this stage do not worry about implementing the points (b) and (c) since these will be covered in the core program, to be given later.

HOW MAY INTERRUPTS BE GENERATED?

There are two major types of interrupt available to the user, namely internally and externally generated.

The internally generated interrupts, on which this article will concentrate,

may then be subdivided into those generated by the user 6522 Versatile Interface Adaptor (VIA) and those generated by other 'events' within the micro.

Interrupts from the 'outside world' are also handled by the user 6522 VIA but will not be covered in this article.

To start with it is the interrupts generated by the occurrence of a particular event, which will be considered. These events are summarised below, but for further details see pages 426 and 465 in the BBC User Guide.

Event Code	Event Enabled
0	O/P buffer empty
1	I/P buffer full
2	Character entering buffer
3	ADC conversion complete
4	Start of Vertical Sync pulse
5	Interval timer crossing zero
6	Escape pressed.

The event code is the number which will be entered into the core program and governs which particular event is enabled. Only events which have been enabled can cause an interrupt. Such events can be enabled and disabled by means of *FX calls (see User Guide as above) as an alternative to the machine code which we shall use. This again will be covered and explained when the core program is set up.

At this stage, we shall not develop our ideas on interrupts further, but simply say that an interrupt is generated upon any of the above 'event' conditions being fulfilled, assuming that the 'event' has been enabled. How to cope with all this is the subject of the next section.

EXAMPLES OF INTERRUPT HANDLING

Before we look at any specific examples, we will look at a skeleton program (the 'core' program) that we can then use as a simple pattern for our programs. The core program, written in 6502 assembler, is in fact presented as a procedure so that it can be used readily in any Basic program. The procedure is called PROCASSEMBLE and is contained in lines 1000 to 1540. The purpose of the procedure is to assemble the machine code into the area of memory reserved by the DIM instruction in line 1010. This is done by calling the procedure in the main program at line 110. Nothing happens until we call the assembled code in line 120.

The core program:

```
  10 REM Program INTRP1
  20 REM Version B1.1
  30 REM Author Trevor Pullen
  40 REM BEEBUG November 1983
  50 REM Program subject to Copyright
 100 MODE 7
 110 PROCassemble(4)
 120 CALL start
 130 END
1000 DEF PROCassemble(ECODE)
1010 DIM EVENTS 200
1020 FOR PASS=0 TO 3 STEP 3
1030 P%=EVENTS
1040 [
1050 OPT PASS
1060 .start
1070 CLD
1080 LDX #(ECODE)        ;event code
1090 LDA #14
1100 JSR &FFF4           ;enable event
1110 LDA #(entry MOD 256)
1120 STA &220            ;lo byte
1130 LDA #(entry DIV 256)
1140 STA &221            ;hi byte
1150 RTS
1160 .entry
1170 PHA:TXA:PHA:TYA:PHA:PHP
1500 PLP:PLA:TAY:PLA:TAX:PLA
1510 RTS
1520 ]
1530 NEXT PASS
1540 ENDPROC
```

Explanation:

There are two separate routines contained in the core program. One, identified by the label START, enables the event specified (depending on the value assigned to the variable ECODE when calling the procedure PROCASSEMBLE) and sets the indirection vector to point to the start of the interrupt routine, identified by the label ENTRY. The START routine could be in Basic using *FX14 to enable the required event (in this case *FX14,4)

and the '?' indirection operator to set the vectored address (see User Guide pages 426 and 409).

The interrupt handling routine at ENTRY simply contains, at line 1170, the instructions to save registers A, X and Y together with the processor status byte, and at line 1500, the corresponding instructions which restore all the registers again before returning to the main program. The interrupt handling routine itself is inserted between these two lines. All our examples will be based on this outline program.

PROGRAM 1

Task: To provide a continuously incrementing counter on the screen.

Method: Since a regularly recurring interrupt is required the 'start of vertical sync pulse' seems a logical choice. Thus call PROCASSEMBLE with ECODE set to 4 in the core program. Note that this program may easily be extended, by the addition of a few more counters, to provide a real time clock (see article on Mode 7 Timer in this issue of BEEBUG).

Program:

```
  100 MODE7:?&71=0:VDU28,0,24,39,1

 1180 INC &70     ;increment 1st counter
 1190 LDA #50      ;interval
 1200 CMP &70     ;interval reached?
 1210 BNE finish ;no - exit
 1220 LDA #0      ;reset
 1230 STA &70     ;first counter
 1240 INC &71     ;increment 2nd counter
 1250 LDA #10     ;2nd counter counts to
10
 1260 CMP &71     ;counter reached 10
 1270 BNE display;not yet
 1280 LDA #0      ;reset
 1290 STA &71     ;2nd counter
 1300 .display
 1310 LDA &71      ;get 2nd counter
 1320 ADC #47      ;convert to ASCII
 1330 STA &7C15    ;display on screen
 1340 .finish
```

We have shown above only those lines that are additional to the core program already listed.

Explanation: Upon receipt of each interrupt, and they occur every 1/50th of a second, a counter is incremented. When the value set by "interval" (in the example 50) is reached, a second counter is then incremented. In each case when the limiting count value is reached, either 'interval' or '10', the appropriate counter is zeroed. The final section of the program converts the value in &71 to ASCII and displays it. In the main program, the memory location &71 is initialised to zero (in line 100) before assembling and calling the interrupt routine.

PROGRAM 2

Task: To provide audio feed back upon each keypress.

Method: This time 'event' 2 is being used, so set the 'event code' to 2. There are two additions to the program since a data block to execute a sound command needs to be set up as well.

Program:

```
   60 FOR I=1 TO 8:READ A:?(&6F+I)=A:NE
XT
   70 DATA 1,0,&F1,&FF,&C8,0,1,0
  110 PROCassemble(2)
 1180 ROL A      ;double range
 1190 STA &74    ;save value
 1200 LDX #&70
 1210 LDY #0     ;execute
 1220 LDA #7     ;SOUND
 1230 JSR &FFF1 ;instruction
```

Again, only lines additional to the core program are given above.

Explanation: The data block set up by lines 60 and 70 provides the necessary information for a SOUND command to be executed. In the interrupt routine lines1180 and 1190 then take the ASCII value of the key pressed, double it to increase the resulting frequency range, and store the result into the lo-byte of the frequency data so that the sound generated will partly depend on the key pressed. Lines 1200-1230 then execute the sound command (see also page 461 in the User Guide). Once you have run this program, you will have a musical keyboard for your BBC micro, regardless of what you do, until you press Break (or switch the machine off).

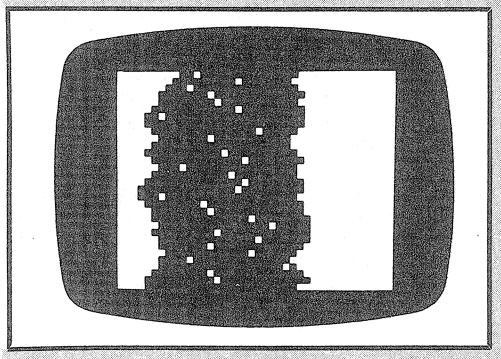
CONCLUSIONS

This article has given a fairly elementary introduction to the concepts and application of interrupts. The main problems in using and applying interrupts are really knowing how to enable the relevant event and then remembering to redirect the required vectors. In a second article, I will explain the use of the 6522 VIA in dealing with both internal and external interrupts.

WRITING A GAME PROGRAM — RAPIDS (16k)

by Alan Webster & Mike Williams

The program listed here is a simple but interesting game of skill which we have called RAPIDS. You have to steer a boat (rather crudely represented and controlled by the usual keys of Z for left and X for right) along a fast flowing river, avoiding the many half submerged rocks as you do so. The river also widens and narrows as you proceed.



The point of this program is to show that an acceptable game can be written in a relative few lines of code. The program as listed contains comment lines to describe the function of each part of the program plus other lines to space out the program and make it more readable. Our original working version was only 19 lines in length so you don't have to be able to write a long program before you can start producing games.

Of course, the game could be improved, by adding to the existing program. You could define another character to represent the boat, and similarly make the rocks, and maybe the banks, more realistic. You could also try putting in a choice of delay times to provide a range from easy (slow) to hard (fast) games. Another feature missing at the moment is any form of scoring and record of high scores. No doubt there are many other changes and enhancements you can think of.

We have listed the program in a form which should make it easy to follow if you want to try and improve it. On the other hand you might like to see how short a program you can write that still produces a reasonable game of some kind. Now there's a challenge!

```
  10 REM Program RAPIDS
  20 REM Version B1.0
  30 REM Author Alan Webster
  40 REM BEEBUG November 1983
  50 REM Program subject to Copyright
  60 :
  70 MODE4
  80 PROCsetupgame
  90 PROCplaygame
 100 END
 110 :
 120 DEF PROCsetupgame
 130 VDU23,255,255,255,255,255,255,255
,255,255
 140 leftbank%=RND(5):rightbank%=40-le
ftbank%
 150 position%=20
 160 ENDPROC
 170 :
 180 DEF PROCplaygame
 190 gameover%=FALSE
 200 REPEAT
 210 :
 220 REM Display river
 230 width%=rightbank%-leftbank%
 240 P$=STRING$(leftbank%,CHR$255)+STR
ING$(width%,CHR$32)+STRING$(40-rightban
k%,CHR$255)
 250 PRINT TAB(0,0)P$
 260 :
 270 REM Place a rock
 280 IFwidth%>10 PRINTTAB(RND(width%)+
leftbank%,0)CHR$255
```

Contd on p. 41

ROM EXTENSION BOARDS REVIEWED

by Philip Le Grand

Many members have expressed interest in ROM expansion boards for the Beeb. We have tested three of the boards now available and present our report to guide you in your choice. Fitting and using all three ROM boards gave our reviewer some unexpected surprises. Now see what we think of these products.

Supplier: WATFORD ELECTRONICS, Cardiff Rd, Watford, Herts.
Price : £35 + VAT + p.& p. (£41.40)

SUPPLIER: SIR COMPUTERS LTD., 91 Whitchurch Rd., Cardiff, S.Glamorgan.
Price : £35 + VAT + p.& p. (£41.40)

SUPPLIER: ATPL LTD., Station Road, Chesterfield, Derby.
Price : £38 + VAT + p.& p. (£44.85)

WHAT IS A ROM BOARD?

As you may know, the BBC micro has a facility for using so-called 'Paged ROMs'. This means that you can 'page out' or de-select the Basic ROM in your machine and select any one of 3 other ROMs or EPROMs plugged into the sockets provided under the keyboard. Both EXMON and WORDWISE operate in this way. An EPROM is plugged in, and called up as an alternative to Basic with commands like *EXMON (or *E) and *WORDWISE (or *W.).

As time goes by, more and more utilities and alternative languages are being offered in ROM or EPROM, and the 3 spare sockets are soon filled - especially since the disc filing system (DFS) occupies one of these sockets. In practice, the BBC micro's operating system can cope with up to 16 paged ROMs, and a number of manufacturers have produced add-on boards to extend the number of physical sockets to that number. Some also allow you to locate RAM (read/write random access memory) as an aid to EPROM development. It should be noted that RAM placed in the paged ROM sockets cannot generally be used for storing Basic programs.

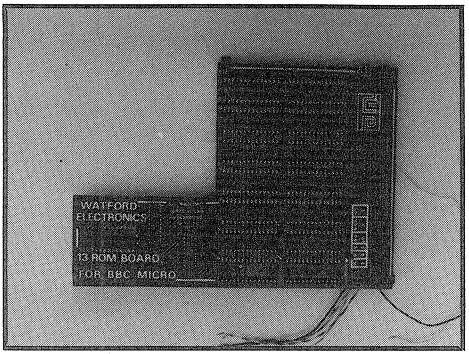
GENERAL COMMENTS

All of the ROM extension boards reviewed were a standard double sided, fibreglass printed circuit board. They all plugged in, and were not screwed down, which may necessitate removing the board for transit of the micro. The board from Sir Computers was the only one without printed identification of the sockets. This identification is useful in determining the order of the sockets, although the markings on the ATPL board were partially obscured by the ROM sockets themselves. They all had header pins on the under-side of the boards, allowing them to stand in the Beeb's ROM, CPU (Central Processor Unit) or ADC (Analogue to Digital Converter) sockets. On two boards, these pins were a thin type, so that inserting them into the Beeb's own sockets, does not over enlarge the holes of the sockets if you want to remove the board at a later date, and replace the original chip - useful if your machine goes wrong under guarantee. The ATPL version supplied, however, had large pins, although we understand that smaller pins will be used on later production models.

For all of our tests, we inserted relevant DIL sockets in the Beeb and put the boards into these, to prevent widening of the holes in the case of the ATPL and also to help raise the boards clear of any power leads or components that were in the way. It was found that all of the boards clashed with the power leads of our Beeb and would not fit properly unless the leads were bent over! This is a delicate operation, since the power lead connectors are easily broken. All of the boards carry between 5 and 10 extra chips for buffering and decoding, and, in ATPL's case, there is a small battery charger option for the battery back-up circuit. The price differences between the boards are small and insignificant.

WATFORD ELECTRONICS

The Watford board is the lightest one reviewed here, and the thinest. To



fit it, you must be prepared to do some very fine soldering inside your BBC micro. The ROM sockets are arranged in a 2 by 7 layout enabling easy removal of any of the ROMs at any time, should you wish to change any. The board can take up to 13 ROMs and, in two of the sockets on the board, you can mix ROM and RAM, allowing a particular ROM to have its own work space. There are link options on the board allowing you to select which type of ROM or RAM you wish to use. The board plugs into the ROM socket labelled IC 100, and any ROM that was occupying that space must be moved out onto the new board. Next, 5 wires from the board must be soldered to IC 76 and IC 14 on the main BBC board. For this job you need a fine tipped, low power (about 18W or lower) soldering iron, 22 SWG solder, and a steady hand. Finally, a socket with two wires on, for enable in and enable out, is plugged in place of link S21. Pushing the board into the vacated ROM socket is quite unnerving since the whole board flexes very easily, and when fitted, it stays curved. The instructions say this is correct, and not to worry!

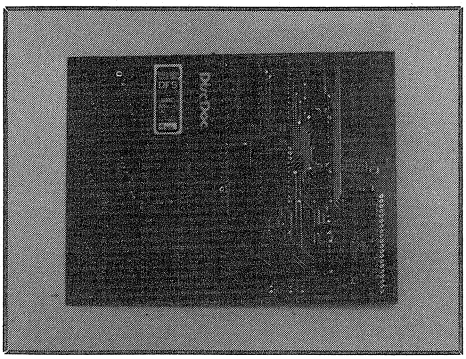
The board has small mounting pillars, although in our machine, it rested on the cassette interface socket with the other supports floating in mid air. When you insert any extra ROMs, the board bends greatly, since the sockets require the ROMs to be pushed home with a substantial force.

In use, the board covers the RAM and video areas which led to over-heating, causing some systems to crash after a few hours of use. The instructions also recommend fitting a fan to the Beeb, although this would mean extensive modification of the case or using it with the lid off.

The documentation supplied consists of a single sheet of instructions and diagrams, which are very clear and concise.

SIR COMPUTERS LTD.

This board is made of thick fibre-glass which will not bend easily, and provides quite substantial support when inserting ROMs. The sockets are arranged in a 3 by 4 layout, allowing 12 extra ROMs to be used, but no RAM, together with the 4 ROM sockets already in the BBC micro. This layout makes it hard to remove a ROM in one of the middle row of sockets if the outer rows are filled unless you have an I.C. insertion tool. We believe that future issues may have a RAM option as well.



To fit this board, no soldering is required. Once again the power connectors inside the Beeb must be bent over to give the new board clearance. Then the CPU must be removed, but to do this, you may find it easier to remove the user 6522 first, then the CPU and finally, the 6522 replaced. The CPU then plugs into the socket on the ROM board. In our tests we plugged the board into an extra 40 pin socket, placed in the Beeb's CPU socket, to prevent any hole enlargement. Finally, link S21 is replaced with a flying lead connector to the ROM board.

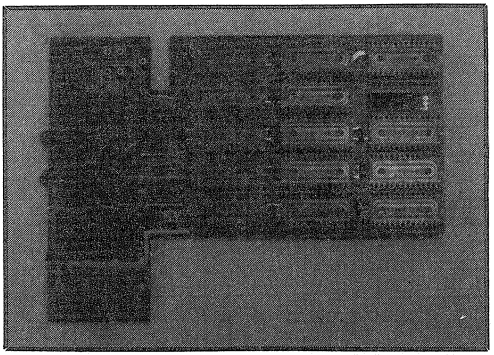
The sockets used are the same type as used on the Watford board, however

insertion of ROMs now does not flex the board, but puts a heavy strain on the CPU socket and could possibly cause damage to the socket and main board if any force was applied. There is no other form of support.

The documentation consisted of a single sheet with no diagrams, although the instructions were clear. This board worked well, but it would have been nice to see some extra sopport, to take the strain off of the CPU socket.

ATPL LTD.

The layout of the sockets on this board also makes it difficult to remove any ROMs in the centre but they are generally easier to get at than on the Sir board. This board is also made from a thick fibreglass which does not flex easily, and was the heaviest of all the boards tested.



To fit the board it is necessary to remove the operating system ROM and the analogue to digital converter chip, which plug into corresponding sockets on the ROM board, and plug the board into the two vacant sockets. This arrangement leads to a very stable board, with the weight evenly distributed. A flying lead is connected to link S21. If during manufacture the solder resist does not cover the board correctly, or for some reason it comes off just above the O.S. ROM socket, then the board could short out on the two diodes immediately above this socket in the Beeb. We used extra mounting sockets to raise the board to gain sufficient clearance. When fitting this board, it was also found necessary to bend a couple of its capacitors flat, so that the keyboard would fit back properly again, though this should not prove necessary as current production versions (issue 3) use much smaller capacitors.

The sockets used are an older type which make it easier to insert and remove the ROMs. The board will accept ROM and RAM chips in various combinations, depending on your selection of the links at the top of the board. The CMOS type RAMs are usable on these boards, enabling the optional battery back-up kit to be added, (thus allowing the memory contents to remain in the RAM even after the Beeb has been switched off). The use of RAM is to allow a program in another ROM to use this as work space or, if you are developing your own programs to be blown into EPROM, you can test it out in its correct operating position and alter it if necessary before committing it to EPROM. No overheating problems occurred, since the board covers the two 6522s, and the serial and video chips, which do not appear to generate as much heat as the Beeb's RAM

The documentation supplied is excellent, consisting of a file of 11 pages of instructions and diagrams, and all the information you need for the various link options. This board was, in our opinion, the best of the three tested, but still not without its faults.

CONCLUSIONS

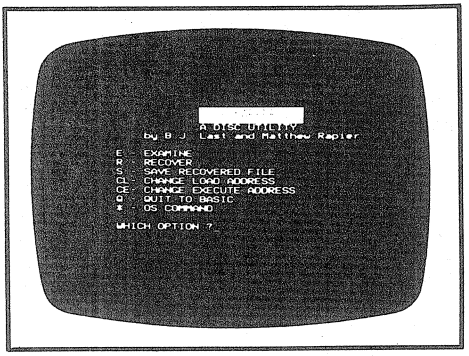
One major difficulty that these and any other expansion boards face is that the Beeb was not designed to accommodate additional boards, although able to cope with up to 16 paged ROMs. These three boards have all adopted slightly different solutions, with varying degrees of success. We believe, from our tests, that the major factors to consider when choosing a ROM board are ease of installation, reliability in operation and the stability of the installed board (particularly important for those who expect to insert and remove ROMs quite frequently). As a general comment, it would appear that most of the problems encountered with these boards would be resolved if they were firmly mounted in a box of their own. Replacing ROMs would then be much easier, and would not involve removing the lid of the Beeb on each occasion.

This will work on O.S. 1.2 and Basics I and II

DISC SNARFER (32k)

by B J Last and Mathew Rapier

SNARFER is a disc utility which amongst other things will allow corrupted or lost files to be retrieved. It makes a useful companion to our recently published Disc Sector Editor, and Disc Sector String Search.



SNARFER is in fact a suite of utility procedures all put together in one program to provide a set of extremely useful commands not available directly in the DFS itself. These commands are roughly equivalent to those available in the new disc utility ROMs appearing on the market. Whilst not as comprehensive as the ROM versions, this program has the same basic capabilities, and is easy to use.

The functions of SNARFER and corresponding commands are:

E - Examine disc sectors
R - Recover disc sectors
S - Save recovered file
CL- Change load address
CE- Change execute address
Q - Quit to Basic
* - OS Command

The program is menu-driven for ease of use, with Escape returning the user to the menu. SNARFER is very useful in file recovery. The Examine option may be used to find a deleted file on the disc and decide on its length. To do this, you need to specify a track number (Ø-39 or Ø-79 depending on disc being used) and a sector number (Ø-9) whereupon the constants of that sector will be displayed on the screen. To recover a file, you will need to locate the beginning and end of the file on disc and calculate the length of the file in bytes (usually number of sectors multiplied by 256). SNARFER requires this to be performed manually, but by using our Disc Sector Editor (BEEBUG Vol2 No3 p27) and Disc Sector String Search (BEEBUG Vol2 No5 p31), this can become a fairly automated process.

With this information you can then proceed to use the Recover routine to load all the sectors belonging to the file into memory (from address &3ØØØ onwards). Having recovered the disc sectors belonging to the file into memory, the file can now be resaved onto disc using the SAVE command. It is still possible that the file may be incomplete or corrupt in some way, but at least it is now accessible in the normal way.

In SNARFER all the * commands can be entered, as in Wordwise, and are passed to the operating system for execution. For example, you can use *DRIVE as normal to select the default drive as the procedures read the current drive number through an OSGBPB call and use this drive for loading sectors.

The commands to change the load and execution addresses of files are also very useful as the normal DFS save file option sets these to zero by default. You can set these values for any file, so that you can alter the load address for *LOAD etc. This is often needed with intelligent file moving programs that will move a file to its load address rather than assuming &EØØ (like Watford Electronics DFS *MLOAD). These commands are particularly simple to implement with OSFILE (a routine at &FFDD in the O.S.), as can be seen from the procedures PROCRELOAD and PROCREEXEC in the program listing.

This utility together with the Disc Sector Editor and Disc Sector String Search, should be on everyone's utilities disc. This program provides a much simpler file recovery system than the sector editor, together with many other useful routines, though the editor is still essential if you want to be able to modify the contents of a sector on disc.

```
10 REM Program SNARFER
20 REM Version B1.0
30 REM Author B.J.Last & BEEBUG
40 REM BEEBUG November 1983
50 REM Program subject to Copyright
60 ONERROR GOTO 1290
70 HIMEM=&3000
80 *OPT1,2
90 @%=4
100 DIMCB%20
110 length=-1:drive=0
120 :
130 REPEAT
140 MODE7
150 FORA=0TO1:PRINTTAB(12);CHR$157;CH
R$135;CHR$129;CHR$141;"SNARFER";SPC(4);
CHR$156:NEXT
160 PRINTSPC(12);"A DISC UTILITY"
170 PRINTSPC(4);"by B.J. Last and Mat
thew Rapier"
180 PRINT''"E - EXAMINE"
190 PRINT"R - RECOVER"
200 PRINT"S - SAVE RECOVERED FILE"
210 PRINT"CL- CHANGE LOAD ADDRESS"
220 PRINT"CE- CHANGE EXECUTE ADDRESS"
230 PRINT"Q - QUIT TO BASIC"
240 PRINT"* - OS COMMAND"
250 PRINT''"WHICH OPTION ?";
260 A$=GET$
270 IFA$="E" THEN PROCEXAMINE
280 IFA$="R" THEN PROCRECOVER
290 IFA$="S"THEN PROCSAVE
300 IFA$="C" THEN PRINT"C";:A$=GET$:I
F A$="L" THEN PROCRELOAD:ELSE IF A$="E"
 THEN PROCREEXEC
310 IFA$="Q" THEN END
320 IFA$="*" THEN PROCOSCLI
330 UNTIL FALSE
340 :
350 DEFPROCEXAMINE
360 PRINT''"EXAMINE"
370 REPEAT
380 PRINT''"Track",;:track=FNEVAL(0)
390 PRINT"Sector",;:sector=FNEVAL(0)
400 PRINT"Error #";~FNREADSEC(&3000)
410 FORN%=0TO255:A%=?(&3000+N%)
420 IFA%<32 OR A%>126 A%=46
430 IF (N% MOD 32)=0 THEN PRINT';~N%,
;
440 VDUA%:NEXT
450 UNTIL FALSE
460 ENDPROC
470 :
480 DEFPROCRECOVER
490 PRINT''"RECOVER"
500 PRINT''"Please type start track,
sector and filelength in bytes."
510 PRINT"Track",;:track=FNEVAL(0):PR
INT"Sector",;:sector=FNEVAL(0):PRINT"Le
ngth",;:length=FNEVAL(0)
520 CLS:PRINTTAB(0,0);"track/sector"
530 address=&3000
540 FORN%=0TOlength+256 STEP256
550 PRINTTAB(0,2);track;TAB(6,2);sect
or;TAB(12,2);"ERROR #";FNREADSEC(addres
s)
560 sector=sector+1:IFsector=10 secto
r=0:track=track+1
570 address=address+256
580 NEXTN%
590 ENDPROC
600 :
610 DEFPROCSAVE
620 PRINT''"SAVE RECOVERED FILE"
630 IFlength=-1 THEN PRINT''"No file
recovered"':PROCRECOVER
640 PRINT''"Enter filename"
650 INPUT $&A00
660 X%=&70:Y%=0:A%=0
670 !&70=&A00
680 !&72=0
690 !&76=0
700 !&7A=&3000
710 !&7E=&3000+length
720 CALL&FFDD
730 ENDPROC
740 :
750 DEFFNEVAL(X)
760 LOCAL Y,A$
770 VDU10,11
780 X=POS:Y=VPOS
790 INPUT A$
800 IF A$="" THEN PRINTTAB(X,Y);:GOTO
790
810 ON ERROR RUN
820 A=EVAL(A$)
830 ON ERROR GOTO 1290
840 =A
850 :
860 DEFFNREADSEC(address)
870 X%=&70:Y%=0:A%=6
880 ?&70=0:!&71=&A00:!&75=1:!&79=0
890 CALL&FFD1:drive=?&A01-48
900 ?CB%=drive:!(CB%+1)=address
910 ?(CB%+5)=3:?(CB%+6)=&53
```

```
  920 ?(CB%+7)=track:?(CB%+8)=sector:?(
CB%+9)=&21
  930 X%=CB%MOD256
  940 Y%=CB%DIV256
  950 A%=&7F
  960 CALL&FFF1
  970 =?(CB%+10)
  980 :
  990 DEFPROCOSCLI
 1000 PRINT''"OS COMMAND"'
 1010 REPEAT
 1020 INPUT "*"$&A00
 1030 X%=0:Y%=&A
 1040 CALL&FFF7
 1050 PRINT"PRESS A KEY":A$=GET$
 1060 UNTILA$<>"*"
 1070 ENDPROC
 1080 :
 1090 DEFPROCRELOAD
 1100 PRINT''"CHANGE LOAD ADDRESS"
 1110 INPUT "FILENAME ",$&A00
 1120 PRINT"ADDRESS ";:address=FNEVAL(0
)
 1130 X%=&70:Y%=0:A%=2
 1140 !&70=&A00
 1150 !&72=address
 1160 CALL&FFDD
 1170 ENDPROC
 1180 :
 1190 DEFPROCREEXEC
 1200 PRINT''"CHANGE EXECUTE ADDRESS"
 1210 INPUT "FILENAME ",$&A00
 1220 PRINT"ADDRESS ";:address=FNEVAL(0
)
 1230 X%=&70:Y%=0:A%=3
 1240 !&70=&A00
 1250 !&76=address
 1260 CALL&FFDD
 1270 ENDPROC
 1280 :
 1290 IF ERR=17 THEN RUN
 1300 PRINT':REPORT
 1310 IF ERR<&BD THEN PRINT " at line "
;ERL:END
 1320 PRINT'"PRESS A KEY"
 1330 A=GET
 1340 RUN
```

Contd from p. 15

brackets enclose the code, DIM statements allocate space, the way P% is used, and so on). Comments may be delimited by a backslash or a semi-colon. When CALL is used, the integer variables A%, B%, C%, D%, E%, F%, H%, and L% can be used to set up the appropriate Z80 registers in the same way that A%,X%,Y% and C% are used in BBC Basic. No facility exists for setting the alternate Z80 register set on entry. Register IX will contain the address of the CALL parameter table.

In BBCBASIC(Z80) it is still possible to use the indirection operators, and two extra statements exist for reading and writing to Z80 I/O port addresses. Reading ports is achieved through an extended version of GET (ie. GET(p) is used), and writing is performed by the PUT statement (ie. PUT(p,v) will write 'v' to port 'p'). These GET and PUT statements are not approved by Acorn and are not guaranteed to remain unchanged!

The major 'awkward' points I found were in the disc I/O statements. The differences are in some sense justified because, let's face it, the two Disc Operating Systems (Acorn DFS and Torch CPN) are different! For example, to obtain a listing of the files on a disc in BBCBASIC(Z80) you have to use *DIR instead of *CAT. This is okay if the system warned you that *CAT (or *.) was illegal - which it doesn't, the system just gets confused!

CONCLUSION

The software should be a welcome addition to those who have gone for the Torch Z80 Disc Pack option. I have certainly found it a delight to use. Its compatibility is excellent, currently permitting well over 47k of program space irrespective of graphics mode. BBCBASIC(Z80) is just what the Torch Disc Pack users need to keep their system compatible with BBC micro software. Its price will probably mean that it is used mainly by business users.

We are grateful to Mr G. Perry of M-TEC Computer Services for the loan of the review software. The price of BBCBASIC(Z80) is £110 + VAT (£126.50 inc.) from M-TEC Computer Services, Ollands Rd., Reepham, Norfolk. NR10 4EL. Tel: Norwich (0603) 870620.

ADVANCED USER GUIDE FOR THE BBC MICROCOMPUTER
ASSEMBLEY LANGUAGE PROGRAMMING ON THE BBC MICRO
DISCOVERING BBC MICRO MACHINE CODE

Reviewed by Mike Williams

We review here three books for the more experienced user of the BBC Micro, particularly those who have progressed to the stage where they want to use machine code. Two of the books claim to introduce the subject of machine code for those new to this form of programming, but how well do they fulfill this task? To start with, we review a book that may well be a classic in the making - now read on.



The Advanced User Guide for the BBC Microcomputer, by A.C. Bray, A.C. Dickens and M.A. Holmes, published by the Cambridge Microcomputer Centre, 512 pages, price £12.95.

This book is surely destined to become the Bible of every dedicated user of the BBC micro. It covers very comprehensively and in considerable detail, every aspect of the machine's operation. It is not primarily intended as a book for the beginner, though there is much that the beginner would find interesting. The book is intended for the experienced user, and moreover one who has some familiarity with 6502 assembler. The purpose of this book is to supplement the BBC User Guide by providing a much more detailed and comprehensive description of 6502 assembly language programming, the operating system and the BBC microcomputer hardware. The book is spiral bound and matches very closely the appearance of the BBC User Guide.

The book begins by surveying the various operating system commands followed by a comprehensive description of the 6502 assembler language that is contained within BBC Basic.

A very substantial chapter provides a detailed description of all FX (OSBYTE) calls and a further chapter similarly covers all the OSWORD calls. One very useful section describes the vectoring mechanism of the Beeb and gives details of each vectored address.

Events and interrupts are clearly described with a number of program examples. There is also a very good section on paged ROMs with details of how to interface a ROM with the operating system.

The remainder of the book, some 90 pages, describes the hardware features of the Beeb, including the video 6845, the ULA, both 6522 VIAs and all the various interfaces. There are also 40 pages of appendices and even a main circuit diagram of the Beeb, courtesy of Acorn.

Many sections of the book are provided with example programs and there are also many clear and useful diagrams. This is a book which the experienced user will delight in dipping into as more and more information is revealed and explained. This book is a must, and is excellent value at the price of £12.95 for over 500 pages packed with facts.

Assembly Language Programming on the BBC Micro, by John Ferguson & Tony Shaw, published by Addison-Wesley, 200

pages, price £7.95.

This book provides an introduction to 6502 assembler programming on the BBC micro. Some knowledge and experience of programming in Basic is assumed but otherwise the book is clearly intended for the newcomer to this form of programming. The text is enlivened by many diagrams and cartoons in an obvious attempt to popularise what has often been considered a difficult subject. However, a lot of detailed information is presented very quickly, perhaps too quickly - the book quite rapidly introduces subroutines, use of the stack, parameter passing - and this before all but the simplest of addressing modes has been covered.

The presentation of information on the page is not as attractive as it might be. The book is printed on poor quality paper such that the printing on the reverse side of the page shows through and distracts from the current page. Some of the diagrams appear very black and heavy, and off-putting to the reader. There are many programs listed by way of example. Unfortunately, these are all reproduced from dot matrix printer originals and reproduction is sometimes less than clear.

The book is excellent in showing how assembler programs are integrated and used within the framework of BBC Basic and there is adequate coverage of operating system calls and other features of the BBC micro such as events and interrupts. Indeed the more advanced sections, shorn of the excesses of the early chapters, are very informative. There are various appendices including a systematic listing of all the assembler mnemonic codes.

This is basically a good book with much useful information, which is somewhat spoilt by poor quality presentation and printing. The attempt to produce a popular beginners book on machine code programming does not seem to have been wholly successful, but nevertheless this is a useful addition to the books on this subject.

Discovering BBC Micro Machine Code, by A.P.Stephenson, published by Granada, 155 pages, price £6.95.

This book is a straightforward introduction to programming in machine code using the 6502 assembler on the BBC micro. Indeed, the approach is rather serious and stolid, with much information presented in the early chapters, and yet little in the way of examples to show how that knowledge might be used. There is a useful description of how to use machine code from within Basic, followed by sections on number representation, registers and addressing modes. The program listings are again from a dot matrix printer but are quite clear as is the presentation of the whole book.

The book, sensibly, does not attempt to cover the more advanced features of machine code such as interfacing or interrupt handling. Instead, it concludes with a useful chapter on programming techniques and guidelines. This is a good book for the serious-minded student of machine code and reasonable value for money. For the complete beginner, the book by Birnbaum (reviewed in BEEBUG Vol.1 No.9) must still be the favourite, while the more experienced will need to look beyond these last two for coverage of more advanced topics and better, more informative examples.

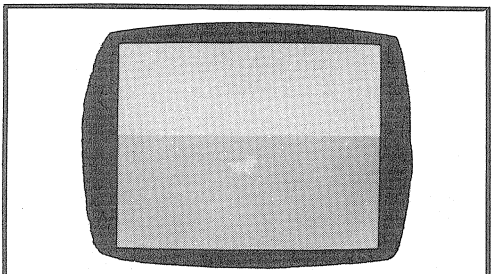
We have been able to include, elsewhere in this issue, a short extract from the Advanced User Guide, reviewed above. This gives details of a new 'MODE 8', and is an example of the valuable information included in this book.

Tested on O.S. 0-1 and 1.2 and on Basics 1 and II

USING THE TELETEXT MODE (Part 2)

by Mike Williams

Last month we presented the first of a series of articles on the Teletext mode which started at the very beginning. We now continue with the second article in the series and introduce Teletext mode graphics.



Teletext Duck by Mike Williams

You should by now be fairly confident about displaying coloured text on the screen and in double height too, using control code 141. We also described how to colour in the background by specifying the colour followed by the control code 157. Hopefully, you also tried out for yourself, the control code 136 for flashing characters. If you're not sure, try typing this into your micro to see the effect produced:

```
PRINT CHR$129 CHR$136 "BEEBUG"
```

You should see 'BEEBUG', in flashing red characters, displayed on the screen. You can also make double height characters flash on and off, but you must include the 136 control code on both lines, just as for the 141 control code, unless of course you only want just the top or bottom half of the letters to flash. The control code 137 can be used to switch from flashing to steady colours, but only needs to be used if flashing characters are to be followed by steady characters on the same line. Try this little example:

```
PRINT CHR$129 CHR$136 "BEEBUG" CHR$137
"MAGAZINE"
```

Notice too, how the 137 control code produces an apparent space between the two words so no extra space is needed.

VDU COMMANDS

At this stage it is helpful to introduce an alternative method of sending Teletext control codes to the screen, namely VDU commands. This avoids the need for the CHR$ function, and can be a much more concise way of handling control codes and other characters. The following statements are, for example, completely equivalent:

```
PRINT CHR$134 CHR$157 CHR$131 CHR$136
"HELLO"
```

and

```
VDU134,157,131,136:PRINT"HELLO"
```

or indeed

```
VDU134,157,131,136,72,69,76,76,79,13,10
```

The codes 72 to 79 are the ASCII codes for the letters in 'HELLO' and the 13 and 10 are the codes for <return><line feed>, which are produced automatically by the PRINT statement unless terminated by a ';' or a ','. To conclude this section, here is a short program which combines all the Teletext mode features we have covered so far.

```
100 REM Program BEEBUG5
110 REM Version B1.1
120 REM BEEBUG November 1983
130 MODE 7
140 ON ERROR GOTO 260
150 FOR Y=0 TO 23
160 VDU 131,157,132,141,13,10
170 NEXT Y
180 VDU28,4,24,39,0
190 VDU23,1,0;0;0;0;
200 PROCmsg2(CHR$(136)+"BEEBUG",12,4)
210 PROCmsg2("for",14,8)
220 PROCmsg2("the",14,12)
230 PROCmsg2("BBC MICRO",11,16)
240 REPEAT UNTIL FALSE
250 END
260 ON ERROR OFF:MODE7
270 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
280 END
500 DEF PROCmsg2(M$,X%,Y%)
510 PRINT TAB(X%,Y%);M$
520 PRINT TAB(X%,Y%+1);M$
530 ENDPROC
```

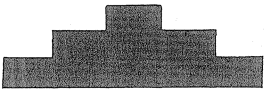
The first part of the program, lines 150 to 170, puts most of the control codes required at the beginning of each line on the screen. These control codes specify yellow (131) background (157), blue (132) double height (141) text. The VDU commands in line 180 and 190 redefine the text window to protect the control codes, and delete the flashing cursor from the screen. The instructions in lines 200 to 220 then display four lines of double height text on the screen, using for convenience, a procedure called PROCmsg2 defined in lines 500 to 530. This just saves having to type every character twice when producing double height characters. The REPEAT UNTIL loop in line 240 avoids the program terminating and hence displaying a '>' character on the screen. The program can be stopped by pressing Escape.

TELETEXT GRAPHICS

Now let's look at Teletext graphics. In many ways this is similar to the text displays we have already seen, in that the effects are again determined by control codes. Try typing in the following instruction:

```
VDU 145,240,252,255,252,240,13,10
```

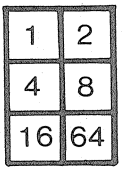
The value of 145 specifies red graphics, while the values of 240, 252 and 255 are the codes of three graphics characters which you should see displayed on the screen. The codes of 13 and 10 at the end of the line are simply <return> and <line feed>. The shape produced should look like this:



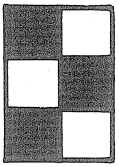
Other values in place of 145 produce different coloured graphics as shown in the table:

Code	Graphics Colour
145	red
146	green
147	yellow
148	blue
149	magenta
150	cyan
151	white

Each graphics character comprises a rectangle consisting of six small squares, coloured or blank. The full range of available graphics characters is listed in the User Guide on pages 488 and 489. You can also work out for yourself the code for any particular graphics character as follows.



Each cell that is coloured contributes a value towards a total to which should also be added 32 + 160. Thus this character has a code of 1 + 8 + 16 + 32 + 160 = 237.

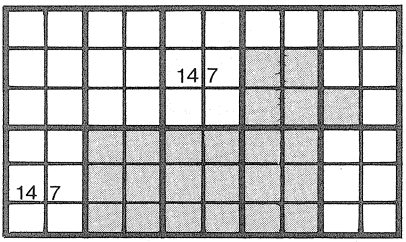


Be careful when referring to the User Guide as the characters are shown with white representing the coloured parts and black the blank parts of each character. It is easy to interpret this the wrong way round.

If you compare the table of graphics characters on pages 488 and 489 of the User Guide with the table of text characters on pages 486 and 487, you will notice that the codes of the graphics characters coincide with those for lower case letters and certain other characters. It is only the choice of control code - graphics or text - which determines what is displayed on the screen. It also means that all the alphanumeric characters (mostly upper case characters) shown in the table of graphics characters, can still be displayed on the screen, even when we are in graphics mode. One difficulty in dealing with graphics characters is that we cannot type them in directly from the keyboard as we can with alphanumeric characters, and this is where the VDU command can be useful.

EXAMPLE OF TELETEXT GRAPHICS

We will now develop a program illustrating the use of teletext graphics, called DUCK1. We need to put together some graphics characters to form the approximate shape of a duck. This is easier to design if we have a grid to work with as shown;



We shall want our duck to be yellow, of course, so we will replace the space characters to the left of the duck by the correct control code for yellow graphics, namely 147. The reason for including the 147 control character here will become clearer later on. We will define our duck as a character string:

```
duck$ = CHR$147 + CHR$255 + CHR$255 +
CHR$255 + CHR$11 + CHR$8 + CHR$8 +
CHR$147 + CHR$252 + CHR$176
```

The definition of our duck starts at the bottom left hand corner with the control code for yellow graphics (147). This is then followed by the codes for the bottom row of graphics characters, namely 255, 255 and 255. We then need to fill in the top row and so codes 11,8 and 8 move up one line, and left two positions, so that the top line is in the right place (see list of codes on page 378 of the User Guide). The top row of our duck consists of the graphics codes 147, 252, 176.

If you type this line in immediate mode you can then display a yellow duck anywhere on the screen by typing, for example:

```
PRINT TAB(10,12);duck$
```

Unless you cleared the screen first, using CLS, then your duck may not be very clear. We want to display our duck against a background of a cyan sky above a blue lake. Setting up background colours like this is most easily done by placing the appropriate control codes down the left side of the screen, a method we have used twice before in this series. The complete program looks like this:

```
100 REM Program DUCK1
110 REM Version B1.0
120 REM BEEBUG November 1983
130 MODE 7
140 ON ERROR GOTO 320
150 FOR Y%=0 TO 10
160 PRINT CHR$(150) CHR$(157)
170 NEXT Y%
180 FOR Y%=11 TO 23
190 PRINT CHR$(148) CHR$(157)
200 NEXT Y%
210 VDU 28,2,24,39,0
220 VDU23,1,0;0;0;0;
230 duck$=CHR$(147)+CHR$(255)+CHR$(25
5)+CHR$(255)+CHR$(11)+CHR$(8)+CHR$(8)+C
HR$(147)+CHR$(252)+CHR$(176)
270 PRINT TAB(15,14);duck$
300 REPEAT UNTIL FALSE
310 END
320 ON ERROR OFF:MODE 7
330 IF  ERR<>17 THEN REPORT:PRINT" at
 line ";ERL
340 END
```

Lines 150 to 200 establish the background colours. Line 210 re-defines the text window to protect the control codes already set up (we have used this technique before too), while the duck is defined, as already described, in line 230 and displayed near the centre of the screen in line 270.

Now, although this looks nice, it's not really very exiting. Add these additional lines to the program, replacing any previous lines as necessary, and you should now find the duck swims across the screen from left to right!

```
240 REPEAT
250 CLS
260 FOR X%=0 TO 33
270 PRINT TAB(X%,14);duck$
280 Z=INKEY(30)
290 NEXT X%
300 UNTIL FALSE
```

The movement is achieved by the FOR NEXT loop in lines 260 to 290 which

print the duck in successive positions across the screen. Remember how, when we originally defined our duck, both lines started with the yellow graphics control code (147). These invisible control codes replace the visible graphics characters as the duck is moved across the screen, neatly overwriting part of the previous image of the duck. The INKEY instruction in line 28Ø controls the speed at which the duck moves. You may like to try other modifications to the program, like adding some scenery or some more ducks.

Next month we will look at some of the more clever things we can do with Teletext graphics. In the meantime, go play with your duck !

THE BEEB IN SLOW MOTION

Basics I and II
O.S. 1·2 only **(16k)**
by Alan Webster

If you'd like to know how to slow the BBC micro down, or maybe just see how it works (e.g. how it clears the screen) in slow motion, then try these two pokes to memory:

```
?&FE46=Ø
?&FE45=1
```

These both access the timers in the internal 6522 VIA, and in simple terms they make the Beeb think that time is going faster than it really is! The machine now spends much more of its time checking for keyboard input and other functions linked to the system VIA, and hence much less time on any user activity, which thus appears to take place much more slowly.

If you now print any text to the screen you will be able to see exactly how each character is formed. Likewise, if you try any graphics command, like the triangle filling routine PLOT 85,X,Y you will see just how this is performed. You may well be able to think of other instructions which would be interesting to view in this way. Try typing in this small program:

```
1Ø MODE 7
2Ø FOR A%=&7C0Ø TO &8ØØØ STEP 4
3Ø !A%=RND(65535)
4Ø NEXT
5Ø ?&FE45=1:?&FE46=Ø
6Ø MODE 2
7Ø PRINT "Finished"
8Ø RUN
```

This shows how the screen is cleared as a result of selecting mode 2, and then how the letters of the word "Finished" are formed. The instructions in lines 2Ø to 4Ø fill the memory mapped screen area with random characters so that there is something to clear when mode 2 is selected, not just a blank screen. Changing the value assigned to &FE46 (in the range Ø to 14) changes the apparent speed of operation.

The simplest way of restoring your Beeb to its normal speed is to press Break (anything else takes so long!). Unfortunately, we have not so far discovered any similar way of speeding up the Beeb, though it is quite fast anyway.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

WORDWISE AND PROGRAM EDITING - W.C.Corbett

You can edit programs with WORDWISE using the option to save the program first, but sometimes on using *EXEC to reload the program after editing, various errors are reported. These are often caused by the line length setting, default 7Ø, which sends a <return> to the computer before the line has been completed. The computer returns the prompt and continues accepting the spooled data, only to give an error when the next <return> is encountered.

It is, therefore, best to keep the program lines short, or set the line length in WORDWISE to a high value.

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

CROSSWORD FRILLS

Dear sir,

From the beginning I have been impressed by the no-frills approach of BEEBUG, keeping very close to the BBC. But now the CROSSWORD has got its toe in, and perhaps there are other peripheral diversions to follow. I see that it is not just a holiday special from the request for more. If this sort of activity is required by other members it will have to be, unless they can be persuaded to get it from the numerous other available sources.

There is a compromise which could also be extended to the rest of the editorial material. We now have the magazine plus a separate advertising supplement. Would it be possible to transfer to the supplement everything of a transient nature, or not applicable to the computer? This would improve the readability of the magazine, and increase the use of the supplement, thereby perhaps drawing more attention to the advertisements.

J.S.Chadwick

P.S. I haven't even managed to do the crossword yet.

Reply: To reassure Mr Chadwick and others who may have similar views, we certainly don't intend to turn BEEBUG into a puzzle magazine, and we do take the point about the supplement which is where any future crosswords will appear. On the other hand our readers have very varied tastes and our puzzles etc have always proved very popular.

COMPUTER WALLPAPER

Dear Sir,

I have a 12 year old son who is very interested in computers, and I am in the process of redecorating his bedroom, but as yet, have found nothing in the line of computer type wallpaper etc. He has his own BBC Microcomputer and I was wondering if you have anything in the way of posters, or advertising material that might be suitable.

Mrs C.E.Coombs

Reply: What an interesting idea - though the thought of walls and walls of computer printout makes the mind boggle! We hope that you can make use of the BEEBUG poster mailed to you; though these are in very short supply, and we cannot normally mail them out on request.

PRINTERS GO SLOW

Dear Sir,

A few weeks ago I became the proud owner of an EPSON FX-80 printer. Now, if you read the EPSON specification and all the laudable reviews one salient and outstanding feature is impressed upon you, its printing speed of 160 cps. Incredible I can hear you say. Read on and you will see that "gullible" would seem more appropriate.

Over a weekend I listed out one of my more ambitious programs. It took a mere 270 seconds to print some 350 lines of Basic, about 16753 characters. Now if you're numerate enough you may be seeing the point of my tale, as this is a rate of 62 cps. Suspicious - you bet I was. I then did a series of timed print runs and the best I could achieve was 97 cps.

"So where is the promised Eldorado of 160 cps?" I asked the marketing man at EPSON UK, reeling under the shock of his admission that my tests were producing perfectly normal results. "Ah well!", he said "It does print at 160 cps IN A STRAIGHT LINE." "If you print at 80 characters and DON'T PRINT A CARRIAGE RETURN OR LINEFEED, and time that, it will take half a second."

Now I realise that everybody else must be printing out just single lines of text at these fabulous speeds and then pasting them together. And here's me getting my kicks from seeing the printhead feeding paper and doing carriage returns.

John Richardson

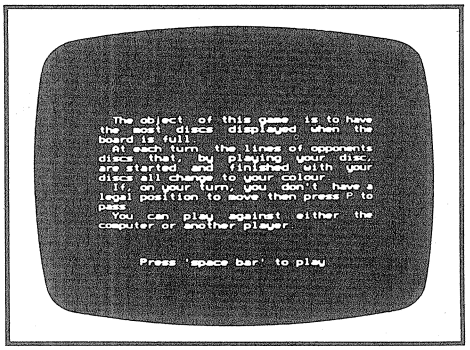
Reply: Although modern bi-directional logic-seeking printers help, many printer manufacturers do quote literally a printing speed rather than an overall throughput speed. Generally, the greater the number of characters per line, the nearer the overall speed will be to that quoted.

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

REVERSI (32k)

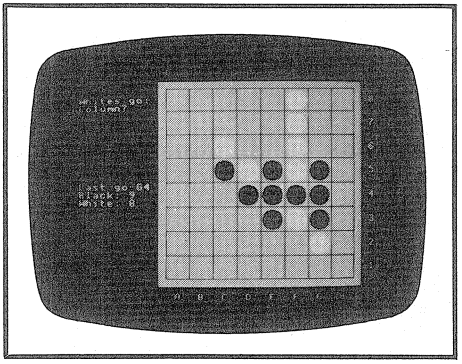
by John Webb

The game of REVERSI (sometimes known as OTHELLO) is played on a conventional Draughts board, with two players. This version is well writen and allows you to play either against the computer or against another player, using the computer to record the moves. You can also choose one of two levels of skill when playing against the computer.



We think you will enjoy playing this well constructed version of a popular game.

Note - to use this program from disc you will need to set PAGE to &1100 before loading and running the program, or use our Disc Program Relocater from BEEBUG Vol.2 No.2. See also BEEBUG Vol.2 No.4 for further information on relocating large programs on disc systems.



In this version of REVERSI the first two moves are always made for you. At the start, you have the choice of colour and whether to play first or second. From then on you must always place one of your counters on the board in such a way that a line of your opponent's counters then has one of your counters at both ends. The surrounded counters are now reversed so that they change to the colour of your own pieces. Of course, you may well be able to place a single counter that captures more than a just a single row of your opponent's pieces.

If on your turn, you are not able to move to a legal position, then you must pass, by pressing P. The game ends when nobody can place any more counters on the board (usually when the board is full), and the winner is the one who has the most counters. The program maintains the current score on the screen throughout the game.

Although a relatively simple game to understand, REVERSI is a game which requires considerable skill to play well. In this version you are able to choose one of two playing levels at the start of each game. Beginners will find even level one quite challenging.

```
10 REM Program REVERSI
20 REM Version B1.1
30 REM Author J.Webb
40 REM BEEBUG November 1983
50 REM Program subject to Copyright
70 MODE7
80 DIMposition%(100)
90 PROCcircles
100 PROCinstructions
110 REPEAT:PROCinit
120 MODE1
130 PROCsetupboard
140 PROCfindplayers
150 REPEAT
160 REPEAT:PROCgo
170 PROCcompute
180 PROClookatgo
```

```
  190 UNTILillegal%=2
  200 PROCupdatescore
  210 turn%=turn%+1:UNTILturn%=60
  220 PROCwin:UNTILwait$<>CHR$(32)
  230  *FX15,1
  250 MODE7:END
  260 DEFPROCinit
  270 FORpos%=1TO100:position%(pos%)=0:
NEXT
  280 turn%=0:value%=1:comp%=0:go%=0:ed
ge%=0:miss%=1
  290 ENDPROC
  300 DEFPROCsetupboard
  310 COLOUR1:VDU28,0,31,10,0,23;8202;0
;0;0;
  320 VDU4
  330 VDU19,2,0,0,0,0,5:GCOL0,2
  340 MOVE352,960:MOVE352,128:PLOT85,11
84,960:PLOT85,1184,128
  350 MOVE416,96:PRINT"A  B  C  D  E  F
  G  H"
  360 FORY%=160TO928STEP96
  370 GCOL0,0:MOVE384,Y%:DRAW1152,Y%:MO
VE1216,Y%+64:GCOL0,2:
  380 IFY%=928THEN390ELSEPRINT;(Y%-160)
/96+1
  390 NEXT
  400 GCOL0,0:FORX%=384TO1152STEP96
  410 MOVEX%,160:DRAWX%,928:NEXT
  420 VDU19,2,2,0,0,0
  430  *FX15,1
  440 RESTORE2090
  450 FORposition%=1TO4:READpos%,val%,C
%:position%(pos%)=val%:GCOL0,C%:PROCfin
drowandcol(pos%):PROCcounter:NEXT
  460 ENDPROC
  470 DEFPROCfindplayers
  480 VDU4:PRINTTAB(0,10)"Do you wantto
 play thecomputer?"'"(Y or N)"
  490  *FX15
  500 REPEAT:wait$=GET$:UNTILINSTR("YNy
n",wait$)>0:SOUND&11,2,40,4:PROCclear(1
0)
  510 IFwait$<>"Y"ANDwait$<>"y"THENENDP
ROC
  520 PRINTTAB(0,10)"Do you wanta  hard
  oreasy game?"'"  (H or E)"
  530  *FX15
  540 REPEAT:wait$=GET$:UNTILINSTR("HhE
e",wait$)>0:SOUND&11,2,40,4:PROCclear(1
0):IFwait$="H"ORwait$="h"THENmiss%=0
  550 go%=1:PRINTTAB(0,10)"Do you wantt
o go first(Y or N)?"
  560  *FX15
  570 REPEAT:wait$=GET$:UNTILINSTR("YNy
n",wait$)>0:SOUND&11,2,40,4:PROCclear(1
0)
  580 IFwait$="Y"ORwait$="y"THENENDPROC
  590 go%=2:ENDPROC
  600 DEFPROCcircles
  610 VDU23,224,0,0,0,3,7,15,31,31,23,2
25,0,0,126,255,255,255,255,255,23,226,0
,0,0,192,224,240,248,248,23,227,31,63,6
3,63,63,63,63,31
  620 VDU23,228,255,255,255,255,255,255
,255,255,23,229,248,252,252,252,252,252
,252,248,23,230,31,31,15,7,3,0,0,0,23,2
31,255,255,255,255,255,126,0,0
  630 VDU23,232,248,248,240,224,192,0,0
,0
  640 C1$=CHR$224+CHR$225+CHR$226:C2$=C
HR$227+CHR$228+CHR$229:C3$=CHR$230+CHR$
231+CHR$232
  650 circle$=C1$+CHR$8+CHR$8+CHR$8+CHR
$10+C2$+CHR$8+CHR$8+CHR$8+CHR$10+C3$
  660 ENVELOPE1,4,0,0,0,0,0,0,121,-10,-
5,-2,120,120:ENVELOPE2,16,4,-8,-4,16,16
,32,64,64,-64,-64,128,0
  670 ENDPROC
  680 DEFPROCclear(tab%):PRINTTAB(0,tab
%)SPC(42):ENDPROC
  690 DEFPROCgo
  700 VDU4:pass%=0:COLOUR2
  710 IFgo%<2THEN740
  720 PROCclear(4):go%=1:PRINTTAB(0,4)"
Computing":IFvalue%=1THENvalue%=2:GCOL0
,0:ELSEGCOL0,3:value%=1
  730 ENDPROC
  740 IFvalue%=1THENPRINTTAB(0,4)"Black
's go:":value%=2:GCOL0,0:ELSEPRINTTAB(0
,4)"White's go:":GCOL0,3:value%=1
  750 IFgo%>0THENgo%=2
  760 REPEAT:REPEAT:PROCclear(5):PRINTT
AB(0,5)"Column? ";
  770 col%=GETAND223:UNTIL(col%>64ANDco
l%<73)ORcol%=80:PRINTCHR$(col%):col%=co
l%-64:IFcol%=16THEN800
  780 PRINTTAB(0,6)"Row? ";
  790 row%=GET:row%=row%-48:PRINT;row%
  800 UNTIL(row%>0ANDrow%<9)ORcol%=16
  810 VDU5:pos%=col%*10+row%+1
  820 ENDPROC
  830 DEFPROCcounter
  840 SOUND&11,1,94,12
  850 VDU5:MOVE291+96*col%,159+96*row%:
PRINT;circle$
  860 TIME=0:REPEATUNTILTIME>50
  870 ENDPROC
  880 DEFPROCillegal
  890 VDU4:SOUND&10,2,70,25
  900 PROCclear(5)
  910 PRINTTAB(0,5)"Illegal"'" move"
  920 IFvalue%=1THENvalue%=2ELSEvalue%=
1
  930 IFgo%=2THENgo%=1
  940 TIME=0:REPEATUNTILTIME>200
  950 ENDPROC
  960 DEFPROCweigh
  970 weight%=0:IFposition%(pos%)>0ENDP
ROC
```

```
  980 RESTORE2100
  990 FORB%=1TO8:READC%:A%=pos%
 1000 IFedge%<>0THENC%=edge%:B%=8
 1010 REPEAT:IFA%=pos%THEN1060
 1020 IFposition%(A%)=0THENA%=199:GOTO1
060
 1030 IFposition%(A%)=value%ANDposition
%(A%-C%)<>value%ANDposition%(A%-C%)>0TH
ENillegal%=1:weight%=weight%+ABS((pos%-
A%)/C%)-1:PROCcheckedge:PROCedge:A%=199
:GOTO1050
 1040 IFposition%(A%)=value%ANDposition
%(A%-C%)=0THENA%=199
 1050 IFweight%>bestgo%THENbestgo%=weig
ht%:compgo%=pos%
 1060 A%=A%+C%:UNTILA%>100ORA%<1
 1070 IFillegal%=1ANDcomp%<>2ANDcol%<>1
6THENA%=pos%:PROCchangecolour
 1080 NEXT:ENDPROC
 1090 DEFPROClookatgo
 1100 black%=0:white%=0:IFcol%=16ANDpas
s%=0THEN1150
 1110 IFpass%=1THENillegal%=2:turn%=tur
n%-1:GOTO1150
 1120 illegal%=0
 1130 IFposition%(pos%)>0THENPROCillega
l:ENDPROC
 1140 PROCweigh
 1150 FORA%=1TO100:IFposition%(A%)=3THE
Nposition%(A%)=value%
 1160 IFposition%(A%)=2THENblack%=black
%+1
 1170 IFposition%(A%)=1THENwhite%=white
%+1
 1180 NEXT
 1190 IFcol%=16ANDillegal%=0THENillegal
%=2:turn%=turn%-1
 1200 IFillegal%<>2THENPROCillegal
 1210 IFblack%=0ORwhite%=0THENturn%=59
 1220 ENDPROC
 1230 DEFPROCfindrowandcol(posit%)
 1240 col%=posit%DIV10
 1250 row%=(posit%-1)MOD10
 1260 ENDPROC
 1270 DEFPROCchangecolour
 1280 REPEAT:PROCfindrowandcol(A%)
 1290 IFposition%(A%)<>3THENPROCcounter
 1300 position%(A%)=3:A%=A%+C%:UNTILpos
ition%(A%)=value%:illegal%=2
 1310 ENDPROC
 1320 DEFPROCupdatescore
 1330 VDU4:PRINTTAB(0,15)"Last go-";
 1340 IFcol%=16THENPRINT"P ":K%=K%+1:EL
SEPROCfindrowandcol(pos%):PRINTCHR$(col
%+64);row%:K%=0
 1350 col%=0:PRINTTAB(0,16)"Black: ";bl
ack%;" "
 1360 PRINTTAB(0,17)"White: ";white%;"
":IFK%=2THENturn%=59
 1370 ENDPROC
 1380 DEFPROCinstructions
 1390 VDU23;8202;0;0;0;:SOUND&11,2,80,2
55
 1400 PRINTTAB(14,3);CHR$130;CHR$141;"R
EVERSI"
 1410 PRINTTAB(14,4);CHR$130;CHR$141;"R
EVERSI"
 1420 PRINT''CHR$131;" The object of
this game is to have"'CHR$131;"the mo
st discs displayed when the"'CHR$13
1;"board is full."
 1430 PRINTCHR$131;" After a disc is p
layed, all of your "'CHR$131;"opponents
 discs trapped between the"'CHR$131;"di
sc just played and another disc of"'CHR
$131;"yours are reversed to your colour
."
 1440 PRINTCHR$131;" If, on your turn,
 you don't have a"'CHR$131;"legal pos
ition to move to then press";CHR$136;"P
";CHR$137;CHR$131;"to pass."
 1450 PRINTCHR$131;" You can play a
gainst either the"'CHR$131;"computer
or another player."
 1460 PRINT'''TAB(6);CHR$134;"Press 'sp
ace bar' to play."
 1470  *FX15,1
 1480 REPEAT:wait$=GET$:UNTILwait$=" "
 1490 ENDPROC
 1500 DEFPROCwin
 1510 VDU5:GCOL0,3:MOVE600,32:SOUND&11,
2,80,255
 1520 IFblack%>white%PRINT"Black wins!"
:GOTO1540
 1530 IFwhite%>black%PRINT"White wins!"
ELSEPRINT"Close game!"
 1540 VDU4:COLOUR1:PRINTTAB(0,23)"Press
 spacebar to play  again"
 1550  *FX15
 1560 wait$=GET$
 1570 ENDPROC
 1580 DEFPROCedge:IFG%<>0THENENDPROC
 1590 RESTORE2040:Q%=0:REPEAT:Q%=Q%+1:R
EADcheck%:UNTILcheck%=pos%
 1600 RESTORE2040:FORP%=1TOQ%+1:READche
ck%
 1610 IFP%=Q%-1ANDposition%(check%)>0AN
Dposition%(check%)<>value%THENweight%=0
 1620 IFP%=Q%+1ANDposition%(check%)>0AN
Dposition%(check%)<>value%THENweight%=0
 1630 NEXT
 1640 ENDPROC
 1650 DEFPROCcheckedge:IFedge%=0THENEND
PROC
 1660 IFposition%(pos%-edge%)>0ANDposit
ion%(pos%-edge%)<>value%THENweight%=0
 1670 IFposition%(A%+edge%)>0ANDpositio
n%(A%+edge%)<>value%THENweight%=0
 1680 ENDPROC
```

```
 1690 DEFPROCcorner:FORQ%=1TO3:READpos%
:PROCweigh:RESTORE2080:FORS%=0TOR%+Q%:R
EADT%:NEXT:NEXT:ENDPROC
 1700 DEFPROCcompute
 1710 G%=1:bestgo%=0:illegal%=0:IFcol%=
16THEN1740
 1720 IFgo%<>1THENENDPROC
 1730 comp%=2
 1740 RESTORE2000:FORR%=1TO4:READpos%:P
ROCweigh:RESTORE2000:FORS%=1TOR%:READT%
:NEXT:NEXT
 1750 IFbestgo%>0THEN1950
 1760 IFcol%=16THEN1830
 1770 IFturn%<5ORmiss%=1THEN1830
 1780 FORedge%=-1TO1STEP2:RESTORE2010:F
ORR%=1TO12:READpos%:PROCweigh:RESTORE20
10:FORS%=1TOR%:READT%:NEXT:NEXT:NEXT
 1790 FORedge%=-10TO10STEP20:RESTORE202
0:FORR%=1TO12:READpos%:PROCweigh:RESTOR
E2020:FORS%=1TOR%:READT%:NEXT:NEXT:NEXT
 1800 edge%=0:IFbestgo%>0THEN1950
 1810 G%=0:RESTORE2030:FORR%=1TO16:READ
pos%:PROCweigh:RESTORE2030:FORS%=1TOR%:
READT%:NEXT:NEXT
 1820 G%=1:IFbestgo%>0THEN1950
 1830 RESTORE2050:FORR%=1TO12:READpos%:
PROCweigh:RESTORE2050:FORS%=1TOR%:READT
%:NEXT:NEXT
 1840 IFbestgo%>0ANDmiss%=0THEN1950
 1850 RESTORE2060:FORR%=1TO16:READpos%:
PROCweigh:RESTORE2060:FORS%=1TOR%:READT
%:NEXT:NEXT:IFbestgo%>0THEN1950
 1860 RESTORE2070:FORR%=1TO16:READpos%:
PROCweigh:RESTORE2070:FORS%=1TOR%:READT
%:NEXT:NEXT:IFbestgo%>0THEN1950
 1870 IFmiss%=1THEN1940
 1880 RESTORE2080:FORR%=0TO12:READpos%
 1890 IFR%MOD4=0ANDposition%(pos%)=valu
e%THENPROCcorner
 1900 RESTORE2080:FORS%=0TOR%:READT%:NE
XT:NEXT:IFbestgo%>0THEN1950
 1910 RESTORE2080:FORR%=0TO12:READpos%
 1920 IFR%MOD4=0ANDposition%(pos%)>0THE
NPROCcorner
 1930 RESTORE2080:FORS%=0TOR%:READT%:NE
XT:NEXT:IFbestgo%>0THEN1950
 1940 RESTORE2080:FORR%=1TO16:READpos%:
PROCweigh:RESTORE2080:FORS%=1TOR%:READT
%:NEXT:NEXT
 1950 IFcol%=16THENENDPROC
 1960 comp%=1:IFbestgo%>0THENpos%=compg
o%:ENDPROC
 1970 SOUND&11,1,2,12:VDU4:PRINTTAB(0,5
);"Pass - no"'"legal move":TIME=0:REPEA
TUNTILTIME>200
 1980 col%=16:pass%=1
 1990 ENDPROC
 2000 DATA12,19,89,82
 2010 DATA14,15,16,17,84,85,86,87,13,18
,83,88
 2020 DATA32,42,52,62,39,49,59,69,22,72
,29,79
 2030 DATA14,17,39,69,87,84,62,32,15,16
,49,59,85,86,42,52
 2040 DATA13,14,15,16,17,18,29,39,49,59
,69,79,88,87,86,85,84,83,72,62,52,42,32
,22
 2050 DATA37,36,35,34,44,54,64,65,66,67
,57,47
 2060 DATA24,25,26,27,38,48,58,68,77,76
,75,74,63,53,43,33
 2070 DATA32,42,52,62,84,85,86,87,69,59
,49,39,17,16,15,14
 2080 DATA19,18,28,29,89,79,78,88,82,83
,73,72,12,22,23,13
 2090 DATA45,2,0,46,1,3,55,1,3,56,2,0
 2100 DATA1,9,10,11,-1,-9,-10,-11
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

FX CALLS GREATER THAN 166 - C.Marshall

When using FX calls greater than 166, all calls go to the same routine in the O.S. This routine accesses the byte at &190+call number. The format used is
<Old contents> AND Y EOR X
Y returns the original contents and X the contents of the next byte up.
When executing these OSBYTE calls you must set X=0 and Y=255 if you do not want to destroy the existing value of contents.

DELETING LINES - D.Fletcher

When deleting lines from the end of a program it is quickest to delete all but the last line in one go and delete the last line afterwards.

MINUTE PRINT ON AN EPSON - J.L.Ward

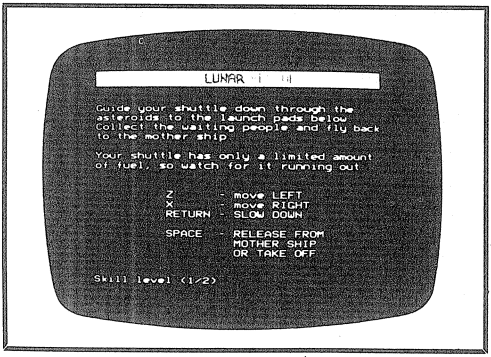
Use the following to produce truly minute print of excellent clarity on an Epson printer. It is achieved by engaging te Superscript mode.
VDU2,1,15,1,27,1,83,1,0,1,27,1,65,1,5
Use VDU1,27,1,84,1,27,1,65,1,12 to reset the printer

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

LUNAR ESCAPE (32k)

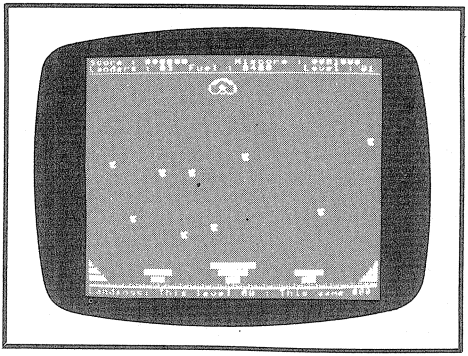
Program by C J Hall

This is an implementation of the arcade game Lunar Rescue. It is entirely in Basic, but despite this is fast enough to be enjoyable. It is fun to play and very addictive.



You are in control of a space shuttle transferring men from the moon's surface. You start off in the main space craft at the top of the screen. You must press the space bar to release from the main craft and drop to the lunar surface. You then dodge the asteroid storm (which gets thicker as you complete each screen) with the use of your steering and the braking rocket. You land on one of the block shaped landing pads where you collect a passenger. You must then press the space bar to take off from the pad again. The landing pad is destroyed in the flame from your thrusters as you take off and, dodging the asteroids, you fly back to the ship. As you approach the ship, its cargo doors will open and you must fly in and disembark the man before loading some more fuel and departing to rescue another person. You repeatedly fly down to the surface until all the landing pads are destroyed.

The program is long and complicated. Take great care when entering the program as debugging is very difficult because of the complicated interactions between the various routines. All the important strings defined at the beginning have been altered in the program to avoid problems with spaces etc. This game is well worth the effort of typing it in. Do not be put off by the concept of an arcade game in Basic, this program runs as fast as needed in all except the highest levels.



```
  10 REM Program LUNAR ESCAPE
  20 REM Version B1.26
  30 REM Author C.J.Hall
  40 REM and BEEBUG
  50 REM BEEBUG November 1983
  60 REM Program subject to Copyright
  70 ON ERROR GOTO 3300
  80 MODE7
  90 PROCinst
 100 PROCsetchars
 110 *FX4,1
 120 II%=RND(-TIME)
 130 IFS%<>-1 !&80=1000
 140 PRINT''CHR$136;"Skill level (1/2)
"
 150 REPEAT:Z$=GET$:UNTIL Z$="1"ORZ$="
2"
 160 IF Z$="1" G%=FALSE:AT%=5 ELSE G%=
TRUE:AT%=10
 170  MODE4:VDU23;11,0;0;0;0,19,0,4;0;
 180 *FX15,0
 190 DOCK=20:LL%=1:S%=0:B%=17000:I%=B%
:FI%=400:F%=FI%:X%=0:NS%=FALSE:U%=0:FLA
G1%=FALSE
 200 NL%=3:IN%=FALSE:AN%=FALSE:D%=0:FT
%=TRUE:A%=981434236:SP$=CHR$(32)
 210 W%=16777183:Y%=-66822:Y1%=-842972
9:H%=!&80:V%=FALSE:BBB%=V%:*FX11,0
```

```
  220 CLS:DIMX%(41),Y%(41),IX%(41),IY%(
41):NG%=0
  230 L$=STRING$(40,CHR$238):PRINTTAB(0
,0)"Score : 000000"
  240 PRINTTAB(0,2);L$;:PRINTTAB(20,0)"
Hiscore : 000000"
  250 PRINTTAB(0,1)"Landers : 00";TAB(1
4,1)"Fuel : 0000";TAB(29,1)"Level : 00"
;
  260 PRINTTAB(0,31)"Landings: This lev
el 00 : This game 000";
  270 SEA1$=STRING$(40,CHR$253)
  280 STMP$=STRING$(2,SP$)+CHR$247+CHR$
245+CHR$249+STRING$(2,SP$)+CHR$10+STRIN
G$(7,CHR$8)+SP$+CHR$243+SP$
  290 STM2$=SP$+CHR$244+SP$+CHR$10+STRI
NG$(7,CHR$8)+STRING$(2,SP$)+STRING$(3,C
HR$248)+SP$+SP$
  300 SHIP$=STMP$+SP$+STM2$
  310 Q$=SHIP$:ANY$=STRING$(3,CHR$232)
  320 WHT$=STRING$(2,CHR$234)
  330 FORII%=0TOH%STEP(H%DIV100):PRINTT
AB(37-LEN(STR$(II%)),0);II%
  340 SOUND0,-15,3,1:NEXT
  350 PRINTTAB(37-LEN(STR$(H%)),0);H%
  360 SOUND1,-15,150,4
  370 FORII%=1 TO 3:PRINTTAB(11,1);II%:
SOUND0,-15,3,3:NEXT
  380 FORII%=20TOF%STEP20:PRINTTAB(25-L
EN(STR$(II%)),1);II%:SOUND0,-15,7,1:NEXT
  390 SOUND1,-15,150,8:PRINTTAB(38,1)"1
"
  400 MANSHIP$=STMP$+CHR$(255)+STM2$
  410 ENTER$=STMP$+SP$+SP$+CHR$244+SP$
  420 PRINTTAB(0,30);SEA1$;
  430 PRINTTAB(0,29);:VDU234,234,232:PR
INTSPC(34);:VDU233,234,234
  440 PRINTTAB(0,28);:VDU234,232:PRINTS
PC(36);:VDU233,234
  450 PRINTTAB(0,27);CHR$232;SPC(38);CH
R$233;
  460 PRINTTAB(9,29);WHT$;SPC(8);WHT$;S
PC(8);WHT$;
  470 PRINTTAB(8,28);WHT$;WHT$;SPC(6);W
HT$;WHT$;SPC(6);WHT$;WHT$;
  480 PRINTTAB(17,27);WHT$;WHT$;WHT$;
  490 IF IN%=TRUE GOTO 1620
  500 L%=RND(25)+5
  510 M%=3
  520 PRINTTAB(L%,M%);SHIP$
  530 N%=RND(3)-2:IF N%=0 GOTO 530
  540 J%=0:K%=0
  550 TIME=0
  560 PROCsetastros
  570 PROCdispastros
  575 BBB%=FALSE
  580 REPEAT
  590 PROCmaninship
  600 PROCmoveastros
  620 UNTIL V%=TRUE
  630 REPEAT
  640 PROCmoveman
  650 PROCmoveastros
  660 PROCmoveship
  670 UNTIL BBB% OR U%=1
  680 IF BBB% THEN BBB%=FALSE:GOTO 580
  690 X%=0
  700 REPEAT
  710 PROCmoveastros
  720 PROCmoveship
  730 UNTIL INKEY(-99)
  740 PRINTTAB(QW%,K%+1)STRING$(6,SP$)
  750 REPEAT
  760 PROCmoveastros
  770 PROCupwego
  780 PROCmoveship
  790 UNTIL BBB% OR DOCK<6:IF BBB% THEN
 BBB%=FALSE:GOTO 580
  810 PROChavewedocked
  820 *FX15,0
  830 GOTO 580
  840 DEFPROCmoveship:T%=L%+N%:IFT%=0 O
R T%=35 N%=-N%:T%=L%+N%
  850 L%=T%:PRINTTAB(L%,M%);SHIP$:ENDPR
OC
  860 DEFPROCmaninship:T%=L%+N%:IFT%=0
OR T%=35 N%=-N%:T%=L%+N%
  870 L%=T%:PRINTTAB(L%,M%);MANSHIP$:IF
 INKEY$(2)<>SP$ ENDPROC
  880 V%=TRUE:PRINTTAB(L%,M%+2);SPC10:P
RINTTAB(L%+3,M%+1);SP$
  890 SOUND1,-15,220,1:J%=L%+3:K%=M%+3
  900 PRINTTAB(J%,K%);CHR$(255):PRINTTA
B(L%,M%);SHIP$:X%=0:ENDPROC
  910 DEFPROCsetastros
  920 FOR Q%=0 TO AT%+1
  930 X%(Q%)=RND(37)+1
  940 Y%(Q%)=RND(18)+6
  950 IX%(Q%)=RND(5)-3:IF IX%(Q%)=0 GOT
O 950
  960 IF G%=FALSE GOTO 980
  970 IY%(Q%)=RND(3)-2
  980 IF Q%MOD5=0 PROCmoveship
  990 NEXT
 1000 ENDPROC
 1010 DEFPROCmoveastros
 1020 IF AT%<10 BG%=1:E%=AT%:GOTO 1050
 1030 IF FT%=TRUE BG%=1:E%=AT% DIV 2:FT
%=FALSE:GOTO 1050
 1040 BG%=AT% DIV 2:E%=AT%:FT%=TRUE
 1050 FOR Z%=BG% TO E%
 1060 T%=X%(Z%)+IX%(Z%)
 1070 R%=Y%(Z%)+IY%(Z%)
 1080 IF T%<0 PRINTTAB(X%(Z%),Y%(Z%))SP
$;:T%=39:X%(Z%)=T%:GOTO1100
 1090 IF T%>39 PRINTTAB(X%(Z%),Y%(Z%))S
P$;:T%=0:X%(Z%)=T%
 1100 IF R%<7 PRINTTAB(X%(Z%),Y%(Z%))SP
$:R%=24:Y%(Z%)=R%:GOTO1140
 1110 IF R%>24 PRINTTAB(X%(Z%),Y%(Z%))S
P$:R%=7:Y%(Z%)=R%:GOTO1140
 1120 PRINTTAB(X%(Z%),Y%(Z%))SP$;
```

```
 1130 Y%(Z%)=Y%(Z%)+IY%(Z%)
 1140 X%(Z%)=X%(Z%)+IX%(Z%)
 1150 PRINTTAB(X%(Z%),Y%(Z%));CHR$(254)
;
 1160 NEXT
 1170 P%=!((K%*40*8)+4+&5800+(J%*8))
 1180 IF P%=A% BBB%=TRUE:PROClosealife
 1190 ENDPROC
 1200 DEFPROCdispastros
 1210 FOR II%=1 TO AT%:PRINTTAB(X%(II%)
,Y%(II%));CHR$(254);:NEXT:ENDPROC
 1220 DEFPROCupwego
 1230 X%=0
 1240 IF INKEY(-98) X%=-1
 1250 IF INKEY(-67) X%=1
 1260 IF INKEY(-74) PROCwaitabit
 1270 IF AN%=TRUE AN%=FALSE:ENDPROC
 1280 PROClessenfuelbyone:J%=J%+X%
 1290 IF BBB%=TRUE THEN ENDPROC
 1300 IF J%<0 J%=0:X%=0
 1310 IF J%>39 J%=39:X%=0
 1320 IF K%<6 DOCK=5:ENDPROC
 1330 IF K%<9 SHIP$=ENTER$:PRINTTAB(L%,
M%+2);SPC(7)
 1340 K%=K%-1
 1350 PRINTTAB(J%-X%,K%+1)SP$;
 1360 P%=!((K%*40*8)+4+&5800+(J%*8))
 1370 PRINTTAB(J%,K%);CHR$(255);
 1380 IF P%=0 ENDPROC
 1390 PROClosealife:BBB%=TRUE:ENDPROC
 1400 DEFPROChavewedocked
 1410 PRINTTAB(L%,M%+2)SPC6
 1420 IF J%<L%+2 OR J%>L%+4 YES=FALSE E
LSE YES=TRUE
 1430 IF YES=FALSE J%=J%-X%:PROCloseali
fe:BBB%=TRUE:ENDPROC
 1440 PRINTTAB(L%,M%);MANSHIP$
 1450 SHIP$=Q$
 1460 J%=L%+3:K%=M%+1
 1470 IF NS%=FALSE oldfuel%=F%:F%=F%+10
0:IF F%>FI% THEN F%=FI%
 1480 IF NS%=FALSE PROCdispfuel(oldfuel
%)
 1490 IF NS%=TRUE NS%=FALSE:PROCnewscre
en
 1500 V%=FALSE:U%=0:DOCK=20
 1510 SHIP$=Q$
 1520 ENDPROC
 1530 DEFPROCnewscreen
 1540 IN%=TRUE
 1550 D%=0
 1560 PROCclearastros
 1570 PRINTTAB(0,12)"Well done ... you
cleared a screen !!"
 1580 PROCnoise:PROCnoise
 1590 PROCbonuscount
 1600 PRINTTAB(0,16)"You have a bonus o
f ":PRINTTAB(20,16);B%;:PRINT" points !
"
 1610 GOTO420
 1620 PROCaddbonus
 1630 AT%=AT%+5
 1640 IF AT% MOD 2=1 AT%=AT%+5
 1650 IF AT%>40 AT%=40
 1660 IF AT%>9 BG%=0:E%=AT% DIV 2
 1670 PROCsetastros
 1680 PROCclearastros
 1690 PROCdispastros
 1700 LL%=LL%+1
 1710 IF LL%=3 NL%=NL%+1:PRINTTAB(12-LE
N(STR$(NL%)),1);NL%:PROCnoise
 1720 SOUND1,-15,150,26
 1730 PRINTTAB(39-LEN(STR$(LL%)),1);LL%
 1740 I%=I%+3000:B%=I%:TIME=0
 1750 FI%=FI%+100
 1760 oldfuel%=F%:F%=FI%
 1770 PROCdispfuel(oldfuel%)
 1780 I%=I%+4000
 1790 B%=I%
 1800 D%=0
 1810 ENDPROC
 1820 DEFPROCclearastros:VDU 28,0,24,39
,7:CLS:VDU 28,0,31,39,0:ENDPROC
 1830 DEFPROCdockingssofar
 1840 PRINTTAB(39-LEN(STR$(TD%)),31);TD
%;:PRINTTAB(22,31);D%;:ENDPROC
 1850 DEFPROCmoveman
 1860 X%=0
 1870 IF INKEY(-98) X%=-1
 1880 IF INKEY(-67) X%=1
 1890 IF INKEY(-74) PROCwaitabit
 1900 IF AN%=TRUE AN%=FALSE:ENDPROC
 1910 J%=J%+X%:PROClessenfuelbyone
 1920 IF BBB%=TRUE THEN ENDPROC
 1930 IF J%<0 J%=0:X%=0
 1940 IF J%>39 J%=39:X%=0
 1950 K%=K%+1
 1960 PRINTTAB(J%-X%,K%-1)SP$;
 1970 P%=!((K%*40*8)+4+&5800+(J%*8))
 1980 PRINTTAB(J%,K%);CHR$(255);
 1990 IF P%=0 ENDPROC
 2000 IF P%=A% OR P%=W% PROClosealife:B
BB%=TRUE:ENDPROC
 2010 IF P%=Y% OR P%=Y1% FLAG1%=2+(P%=Y
%):PROClosealife:BBB%=TRUE:ENDPROC
 2020 PROCblock
 2030 U%=1
 2040 ENDPROC
 2050 DEFPROCblock
 2060 IF J%<13 QW%=6
 2070 IF J%<26 AND J%>12 QW%=17
 2080 IF J%<39 AND J%>25 QW%=26
 2090 PRINTTAB(J%,K%-1);CHR$(255)
 2100 PRINTTAB(J%,K%);CHR$234;
 2110 IF K%=28 SC%=100 ELSE IF K%=29 SC
%=200 ELSE SC%=50
 2120 D%=D%+1
 2130 IF D%=7 NS%=TRUE:D%=0
 2140 TD%=TD%+1
 2150 PROCdockingssofar
 2160 PROCnoise
 2170 S%=S%+SC%:PROCscore
```

```
 2180 K%=K%-1
 2190 ENDPROC
 2200 DEFPROClosealife
 2210 SHIP$=Q$
 2220 NL%=NL%-1
 2230 PRINTTAB(12-LEN(STR$(NL%)),1);NL%
 2240 PRINTTAB(J%,K%);CHR$242
 2250 PROCcrash
 2260 PRINTTAB(J%,K%)SP$
 2270 IF FLAG1% THEN PRINTTAB(J%,K%);CH
R$(231+FLAG1%);:FLAG1%=FALSE
 2280  IF NL%=0 PROCfinish:IF Z$<>"Y" E
ND
 2290 V%=FALSE:U%=0:DOCK=20
 2300 PRINTTAB(0,30);SEA1$;
 2310 J%=L%+3:K%=M%+2
 2320 *FX15,0
 2330 IF NS%=FALSE ENDPROC
 2340 PROCnewscreen:DOCKS=0:NS%=FALSE:E
NDPROC
 2350 DEFPROCfinish
 2370 VDU28,0,31,39,3:CLS:*FX15,0
 2380 PRINTTAB(9,12)"G A M E   O V E R"
 2390 PRINTTAB(0,25)"Do you want to pla
y again (Y/N)":VDU28,0,31,39,0
 2400 IF S%>H% PRINTTAB(0,17)"You got t
he high-score : well done!!":H%=S%:!&80
=H%
 2410 S%=-1:Z$=GET$:IF Z$<>"N" RUN
 2420 *FX4
 2430 VDU22,7:*FX11,40
 2440 ENDPROC
 2450 DEFPROCbonuscount
 2460 B%=I%-TIME:B%=B% DIV 10:IF B%<0 B
%=0:ENDPROC ELSE ENDPROC
 2470 DEFPROCaddbonus:S%=S%+B%:FOR II%=
S%-B% TO S% STEP 10
 2480 PRINTTAB(14-LEN(STR$(II%)),0);II%
:SOUND0,-15,3,1:NEXT
 2490 PRINTTAB(14-LEN(STR$(S%)),0);S%:P
ROCnoise:ENDPROC
 2500 DEFPROClessenfuelbyone
 2510 F%=F%-1
 2520 IF F%=-1 F%=FI%:PROClosealife:PRO
Cdispfuel(10):BBB%=TRUE
 2530 PRINTTAB(21,1)"0000"
 2540 PRINTTAB(25-LEN(STR$(F%)),1);F%;
 2550 ENDPROC
 2560 DEFPROCdispfuel(oldfuel%):PRINTTA
B(21,1)"0000"
 2570 FOR II%=oldfuel% TO F% STEP 10:PR
INTTAB(25-LEN(STR$(II%)),1);II%
 2580 SOUND0,-15,3,3:NEXT:ENDPROC
 2590 DEFPROCwaitabit
 2600 IF F%<11 AN%=FALSE:ENDPROC
 2610 F%=F%-10
 2620 PRINTTAB(21,1)"0000"
 2630 PRINTTAB(25-LEN(STR$(F%)),1);F%
 2640 AN%=TRUE
 2650 ENDPROC
 2660 DEFPROCscore:FOR II%=S%-SC% TO S%
STEP 2
 2670 PRINTTAB(14-LEN(STR$(II%)),0);II%
:SOUND0,-15,3,1:NEXT:ENDPROC
 2680 DEFPROCnoise
 2690 FOR II%=1 TO 6
 2700 SOUND1,-15,5,3
 2710 SOUND1,-15,20,3
 2720 NEXT
 2730 *FX15,1
 2740 TIME =0:REPEAT UNTIL TIME>90
 2750 ENDPROC
 2760 DEFPROCcrash
 2770 FOR II%=-15 TO 0
 2780 SOUND0,II%,3+RND(3),5
 2790 NEXT
 2800 ENDPROC
 2810 DEFPROCinst
 2820 PRINTCHR$134;CHR$157;SPC(12);CHR$
129;CHR$141;"LUNAR ESCAPE"
 2830 PRINTCHR$134;CHR$157;SPC(12);CHR$
129;CHR$141;"LUNAR ESCAPE"
 2840 PRINT''" Guide your shuttle down
through the"
 2850 PRINT" asteroids to the launch pa
ds below."
 2860 PRINT" Collect the waiting people
 and fly back";
 2870 PRINT" to the mother ship."'
 2880 PRINT" Your shuttle has only a li
mited amount"
 2890 PRINT" of fuel, so watch for it r
unning out."''
 2900 PRINTSPC11;"Z      - move LEFT"
 2910 PRINTSPC11;"X      - move RIGHT"
 2920 PRINTSPC11;"RETURN - SLOW DOWN"'
 2930 PRINTSPC11;"SPACE  - RELEASE FROM
"
 2940 PRINTSPC20;"MOTHER SHIP"
 2950 PRINTSPC20;"OR TAKE OFF"
 2960 VDU28,0,24,39,0
 2970 ENDPROC
 3060 DEFPROCsetchars
 3070 VDU23,255,24,60,189,255,102,60,66
,129
 3080 VDU23;11,0;0;0;0
 3090 VDU23,232,&60,&E8,&E8,&E8,&FA,&FA
,&FE,&FF
 3100 VDU23,233,6,&17,&17,&17,&5F,&5F,&
7F,&FF
 3110 VDU23,234;&FFFF;&FFFF;&FFFF;&FFFF
 3120 VDU23,238;255;0;0;0
 3130 VDU23,242,153,90,60,231,231,60,90
,153
 3140 VDU23,254,&4E,&FF,&FF,&FC,&7C,&7F
,&7F,&3A
 3150 VDU23,253,&90,&92,&D2,&DA,&DF,&FF
,&FF,0
 3160 VDU23,243,1,3,3,7,7,3,3,1
 3170 VDU23,244,&80,&C0,&C0,&E0,&E0,&C0
,&C0,&80
```

```
 3180 VDU23,245,&3D,&FF,&FF,&C3,&81,0,0
,0
 3190 VDU23,247,0,3,7,&1F,&3F,&7E,&7C,&
F8
 3200 VDU23,248,&81,&7E,0,0,0,0,0,0
 3210 VDU23,249,0,&C0,&E0,&F8,&FC,&7E,&
3E,&1F
 3220 ENDPROC
```

```
 3300 ON ERROR OFF
 3310 IF ERR=17 THEN PROCfinish ELSE RE
PORT:PRINT" at line ";ERL
 3320 *FX4
 3330 *FX12
 3340 VDU23;11,255;0;0;0
 3350 END
```

Contd from p. 19

```
 290 :
 300 REM Scroll screen down one line
 310 VDU30,11
 320 :
 330 REM Randomly adjust banks
 340 E%=RND(3)-2:J%=E%+leftbank%
 350 IF J%>0 AND J%<(rightbank%-1) THE
N leftbank%=J%
 360 F%=RND(3)-2:K%=F%+rightbank%
 370 IF K%<37 AND K%>(leftbank%+1) THE
N rightbank%=K%
 380 :
 390 REM Check for boat movement
 400 position%=position%+(INKEY-98)-(I
NKEY-67)
 410 :
 420 REM Display boat
 430 PRINTTAB(position%,31);SPC1;TAB(p
osition%,30);"^"
 440 :
 450 REM Check for crash
 460 IFPOINT((position%)*32,76)<>0 THE
N gameover%=TRUE
 470 UNTIL gameover%
 480 VDU7:*FX15,1
 490 ENDPROC
```

POINTS ARISING

Illusions - Vol.2 No.5

If you encountered any difficulties in getting this program to work for you, check carefully that you have typed in 'l%' (lower case L) and 1 (figure one), both of which are used in the program and can cause confusion. For this very reason we normally try to avoid this possible source of confusion. In all respects however, as far as we are aware, the program is bug free.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

THE FOUR WAYS OF USING FUNCTION KEYS - Paul Beverley

1) On their own either as normal *FX225,1 or disabled with *FX225,0 or to produce characters *FX225,n
2) With SHIFT *FX226,n
3) With CTRL *FX227,n
4) With CTRL and SHIFT *FX228,n

Where 'n' is the base number for the ASCII codes generated.

DOUBLE QUOTES - Alan Baker

To get a " mark inside a piece of text delimited by "..." use two together:

```
PRINT "THIS IS A "" DOUBLE QUOTE"
```

It gives:

```
THIS IS A " DOUBLE QUOTE
```

If you are going to put a double quote in the middle of a string then the string must be delimited with normal quotes and the quote marks entered inside with two together as above.

TAB in WORDWISE - A.W.Rowe

Pressing tab repeatedly at the beginning of the line will not give the correct indentation. Use Tab Space Tab instead.

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.
Thanks are due to David Graham, Sheridan Williams, Adrian Calcraft, Tim Powys-Lybbe, Matthew Rapier, and David Earlam for assistance with this issue.

All reasonable precautions are taken by BEEBUG to ensure that the advice and data given to readers are reliable. We cannot, however, guarantee it and we cannot accept legal responsibility for it, neither can we guarantee the products reviewed or advertised.

BEEBUG (c) November 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.

SPECIAL OFFERS FROM



THIS MONTH

WORDWISE Wordprocessor £38

EXMON Machine Code Monitor Cassette £6.90 EPROM £14.50

MASTERFILE File Management Program on Cassette £6.75
on Disc £11.00 and £12.00

OMEGA £4.00

SWARMER £4.00

NEW 1.2 OPERATING SYSTEM ROM £5.85

MAGAZINE CASSETTE SUBSCRIPTION £33 P.A.

BEEBUG MAGAZINE BINDERS £3.90

MEMOREX BLANK 5.25 INCH DISCS
40 TRACK £15.50 80 TRACK £26.50

ALIEN DESTROYER £6.00

DISC DOCTOR £30.00

ADVANCED USER GUIDE £12.95

NEW SOFTWARE CLUB FOR BEEBUG MEMBERS
Games from Program Power, Alligata, Superior Software and Software Invasion.

SPECIAL BEEBUG DIARIES WILL SOON BE AVAILABLE TO MEMBERS. FULL DETAILS IN THE NEXT ISSUE OF BEEBUG.

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.

This month's cassette (Vol.2 No.6) includes: SNARFER Disc Retriever, Teletext (Part 2) programs, Mode 7



Clock, Interrupt example programs, Multi-character Definer, RAPIDS game, Demolition, Lunar Rescue game, Reversi game, Mode 8 Display, Fast Sort Routine, Swarmers' Revenge 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.

ISSN 0263-7561