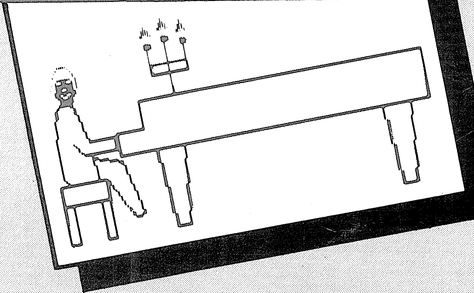
£1.00

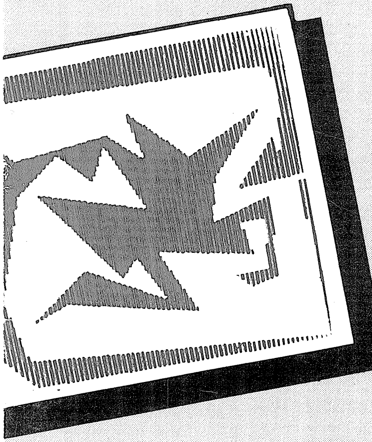
BEEBUG

FOR THE BBC MICRO

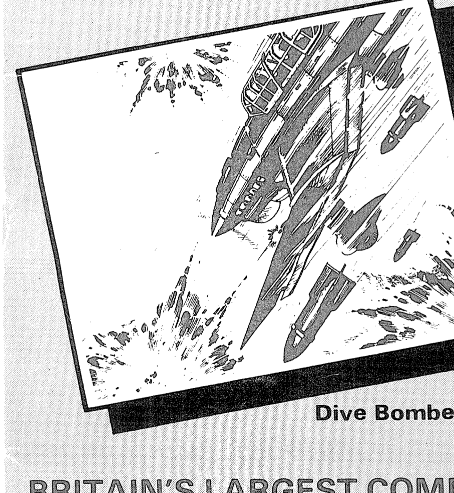
Vol 3 No 3 JULY 1984



BEEBUG plays Bach



Fast Colour Fill Routine



Dive Bomber Game

- ○ BEEBUG plays Bach
- ○ Acorn 6502 second processor reviewed
- ○ Multi-screen slide show
- ○ Printer spooler
- ○ Fast colour fill routine
- ○ Viewsheet & Ultracalc reviewed
- ○ Snip-snap game
- ○ Dive bomber game
- ○ And much more

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 25,000

EDITORIAL

REPLY TO THE MICRO USER

It is with some regret that we are forced to take up space in this magazine to reply to an ugly and quite unjustified item in the normally reputable journal 'The Micro User'. A news item in the June 1984 issue of that magazine implied that BEEBUG was what it called a "PSEUDO USER GROUP", and that we had attempted to "muscle in on the user group scene", maintaining that we had been "expelled" by the Association of Computer Clubs (ACC) because we were a commercial group with no amateur status.

In practice, although it is a fine point, we were not expelled; the ACC was reconstituted with new rules in October 1983 and now requires all affiliated clubs to be amateur. Like a number of other previously affiliated clubs BEEBUG does not meet with this condition; though we are informed that we could apply for so-called corporate status of the ACC if we so wished. To say that we have been expelled is to give quite the wrong impression. We have committed no misdemeanour, and are on good terms with members of the ACC executive.

The article also implies that we are a "pseudo user group" because we are not an amateur club. This is patent nonsense. A footballer does not become a psuedo-footballer when he moves from amateur to professional status. Or to take another example, the RAC (Royal Automobile Club - user support for car owners) is not a "pseudo" motoring organisation because it is commercial. In practice it is only by being commercial that the RAC can provide the service that it does.

It is the same with BEEBUG. It is only possible to provide our current level of support for BBC micro users by being commercial - by employing full time editorial, software and production staff etc. At the same time however, we have great respect for the many non-commercial computer clubs and societies; each has its own part to play in supporting microcomputer users.

We understand that the Micro User news item in question was published without the knowledge of the editor Mike Bibby; and we have now received an assurance by Derek Meakin, Managing Director of Database publications, that the records will be "put straight".

NOTICEBOARD

Because of the space taken up by the editorial comment above, this month's Noticeboard items appear on page 29.

BEEBUG MAGAZINE

GENERAL CONTENTS

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

BEEBUG PLAYS BACH (32K)

by Brian Knott

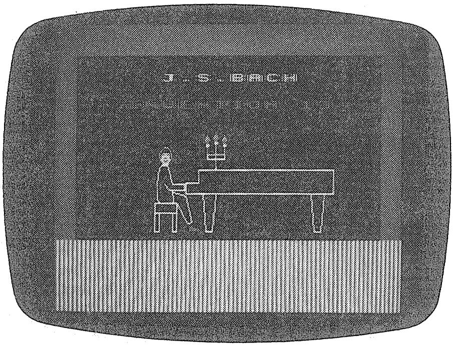
In BEEBUG Vol.2 No.9 we published a program to play Bach's Cantata No. 147, and here we follow that with a graphical, as well as musical, version of Bach's Invention number 13. The author is the winner of the Roman Numeral Brainteaser Competition with the program 'Stonemason' that we also published in the March issue.

Due to the popular response to the piece of Bach we published previously, we present another, this time with a delightful animated screen display. The graphics are most amusing and depict the kind of scene one might expect with a virtuoso performance such as this. The curtains are drawn back to reveal a pianist sitting at a grand piano, and while the music plays he taps his foot, the candles flicker in the candelabra, and from time to time he turns to face the audience.

Once you have typed the program in, taking particular care with the data at the end, and saved it to cassette or disc, you just need to run it and sit back and enjoy the performance.

PROGRAM NOTES
Each note is held as a three character string, with the data for the notes starting at line 1620. The note is read into DD$ in line 190 and then decoded in lines 270 to 290. PP% contains the channel number, which is then passed to P% at line 260. The envelope is determined in line 270, the pitch is set in line 280 and the duration in line 290.

All in all this program is great fun and the effort required to type it in is very well rewarded.



```
   10 REM PROGRAM BACH2
   20 REM VERSION B0.3
   30 REM AUTHOR  B.KNOTT
   40 REM BEEBUG  JULY 1984
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 ON ERROR GOTO 1750
  110 MODE1
  120 VDU23,1,0;0;0;0;
  130 PROCchar
  140 PROCloud
  150 PROCpic
  160 PROCxtra
  170 REM *** MAIN ROUTINE ***
  180 F%=120:G%=120:TIME=12
  190 FORX%=1TOC%:READD$:PP%=VALLEFT$(D
$,1):R$=MID$(D$,2,1):S$=RIGHT$(D$,1)
  200 Y%=RND(10)+10:IFTIME>50 GCOL0,1:M
OVE544,320:VDU127,233:F%=X%:GCOL0,0:TIM
E=0
  210 IFX%=F%+2GCOL0,1:MOVE544,320:VDU1
27,231:GCOL0,0
  220 IF X% MOD 110=0 MOVE444,576:VDU12
7,127,248,249,10,127,127,250,251:G%=X%
  230 IF X%=G%+7 AND RND(2)=1 MOVE444,57
6:VDU127,127,252,253,10,127,127,254,255
  240 IF X%=G%+15 MOVE444,576:VDU127,12
7,244,245,10,127,127,246,247
  250 IFX% MOD Y%=0 MOVE644,614:VDU127,
235:MOVE612,622:VDU127,235:MOVE580,614:
VDU127,235:MOVE612,614:VDU234:MOVE580,6
22:VDU234:MOVE548,614:VDU234:GCOL0,1:MO
VE612,614:VDU235:MOVE580,622:VDU235:MOV
E548,614:VDU235:GCOL0,0
  260 P%=PP%+(253ANDPP%>3)+(253ANDPP%>6)
  270 IFR$="<" Q%=0 ELSEQ%=1
  280 R%=(ASC(R$)-60)*4-11
  290 S%=3*(ASC(S$)-59)
  300 SOUNDP%,Q%,R%,S%
  310 NEXT
  320 MOVE444,576:VDU127,127,248,249,10
,127,127,250,251
  330 VDU26,30,4
  340 END
  350 :
 1000 DEFPROCloud
 1010 REM *** SET VOLUME, ENVELOPE, COL
OURS, WINDOWS ***
 1020 RESTORE1620:C%=579
 1030 REPEAT:PRINTTAB(8,16)"WHAT VOLUME
 LEVEL (1-4) ?":L%=GET-48:UNTIL L%>0 AN
D L%<5:ALA%=16+(L%*22):ENVELOPE1,1,0,0,
0,0,0,0,127,-1,-1,-1,ALA%,0
 1040 VDU26,12,24,0;0;1279;256;:GCOL108
,128:CLG:VDU24,0;260;1279;1023;19,0,1;0
;19,2,0;0;
 1050 GCOL0,129:CLG
 1060 ENDPROC
 1070 :
 1080 DEFPROCchar
 1090 REM *** DEFINE CHARACTERS ***
 1100 VDU23,230,0,0,0,0,15,15,15,15
 1110 VDU23,231,0,0,0,0,0,248,252,252
 1120 VDU23,233,0,0,0,56,248,248,192,0
 1130 VDU23,234,16,16,56,56,124,124,56,
56
 1140 VDU23,235,0,0,16,16,56,124,56,56
 1150 VDU23,240,0,7,8,16,32,32,65,73
 1160 VDU23,241,0,128,96,16,16,16,240,56
 1170 VDU23,242,83,83,75,67,39,31,31,31
 1180 VDU23,243,248,252,224,240,128,224
,224,128
 1190 VDU23,244,0,0,0,3,4,4,8,8
 1200 VDU23,245,0,0,224,24,4,4,2,2
 1210 VDU23,246,17,18,18,35,35,31,15,15
 1220 VDU23,247,60,228,252,252,252,208,
224,224
 1230 VDU23,248,0,7,8,16,32,32,67,86
 1240 VDU23,249,0,128,64,32,16,16,248,96
 1250 VDU23,250,87,87,79,47,46,31,15,15
 1260 VDU23,251,240,240,248,192,224,0,2
24,192
 1270 VDU23,252,0,7,8,16,32,32,79,83
 1280 VDU23,253,0,128,64,32,16,16,200,40
 1290 VDU23,254,95,95,92,63,59,28,15,15
 1300 VDU23,255,232,232,232,240,112,224
,192,192
 1310 ENDPROC
 1320 :
 1330 DEFPROCpic
 1340 REM *** DRAW SCENE ***
 1350 GCOL0,0:MOVE0,256:DRAW1280,256
 1360 MOVE368,352:PLOT1,0,-64:PLOT1,16,
0:PLOT1,0,64:MOVE432,352:PLOT1,0,-64:PL
OT1,16,0:PLOT1,0,64:PLOT1,0,32:PLOT1,-8
0,0:PLOT1,0,-32:PLOT1,80,0
 1370 MOVE480,416:PLOT1,544,0:PLOT1,0,8
0:PLOT1,-496,0:PLOT1,0,-44:PLOT1,-48,0:
PLOT1,0,-36:REM PIANO
 1380 MOVE544,416:PLOT1,12,-128:PLOT1,2
4,0:PLOT1,12,128:MOVE944,416:PLOT1,12,-
128:PLOT1,24,0:PLOT1,12,128:REM  LEGS
```

```
 1390 MOVE592,496:PLOT1,0,76:PLOT0,-12,
40:VDU5,234:PLOT0,-64,-8:VDU234:PLOT0,-
20,-40:PLOT1,0,-32:PLOT1,64,0:PLOT1,0,3
2:PLOT0,-12,40:VDU234:REM CAND
 1400 MOVE384,384:PLOT1,64,0:PLOT1,40,-
68:PLOT1,16,4:PLOT1,-24,88
 1410 PLOT1,-48,12:PLOT1,0,8:PLOT1,4,-4
0:PLOT0,-4,40:PLOT1,-12,0:PLOT1,-16,20:
PLOT0,16,-20:PLOT0,8,0
 1420 PLOT1,48,0:PLOT1,0,20:PLOT1,-42,0
:PLOT1,-20,64
 1430 PLOT1,-20,0:PLOT1,-8,-8:PLOT1,-8,
-54:PLOT1,0,-54:PLOT1,8,-8:REM MAN
 1440 MOVE380,572:VDU240,241,8,8,10,242
,243:REM HEAD
 1450 MOVE480,320:VDU230,231:REM FOOT
 1460 REM *** OPEN CURTAINS ***
 1470 GCOL4,1:FORI%=3TO560STEP4:MOVE640
+I%,260:DRAW640+I%,896:MOVE640-I%,260:D
RAW640-I%,896:NEXT
 1480 ENDPROC
 1490 :
 1500 DEFPROCxtra
 1510 REM *** DETAILS OF SCENE ***
 1520 VDU4,19,0,3;0;:COLOUR130:COLOUR0
 1530 ?&360=15:?&34F=32:PRINTTAB(6,6)"J
.S.BACH"
 1540 COLOUR3:PRINTTAB(4,9)"INVENTION 1
3"
 1550 ?&360=3:?&34F=16
 1560 GCOL0,0:GCOL0,130
 1570 MOVE592,512:PLOT1,0,76:PLOT0,-32,
-8:PLOT1,0,-32:PLOT1,64,0:PLOT1,0,32
 1580 GCOL0,1:MOVE480,320:VDU5,230,231:
GCOL0,0
 1590 MOVE444,576:VDU127,127,244,245,10
,127,127,246,247:MOVE612,614:VDU234:MOV
E580,622:VDU234:MOVE548,614:VDU234:GCOL
0,1:MOVE612,614:VDU235:MOVE580,622:VDU2
35:MOVE548,614:VDU235:GCOL0,0
 1600 ENDPROC
 1610 :
 1620 DATA4F<,5<<,4Y<,5<<,4^<,5R?,1a<,1
`<,1Y<,4`<,5Q=,1c<,4R<,5a=,1M<,4R<,5e<,
1U<,4T<,5]<,1M<,4T<,5e<,1W<,4^<,5U=,1Y<
,4^<,5R=,1a<,4`<,5Q<,1Y<,4`<,5M<,1c<,4R
<,5a=,1M<,4R<,5^<,1U<,1T<,1M<,1T<,1W<,4
<<,5U=,1e<,4a<
 1630 DATA5R<,1e<,4^<,5U<,1a<,4Y<,5R=,1
\<,4W<,5Z=,1R<,4N<,5^<,1R<,4K<,5c<,1N<,
4F<,5f>,1I<,1H<,0<;,1c<,4`<,5K<,1c<,4\<
,5P<,1`<,4W<,5T>,1Z<,1Y<,0<;,1P<,4M<,5\
<,1P<,4I<,5a<,1M<,4D<,5e>,1H<,1F<,1a<,4
^<,5I<,1a<,4K<
 1640 DATA5Z<,1N<,4H<,5c>,1K<,1D<,1`<,4
\<,5H<,1`<,4I<,5Y<,1M<,4F<,5a>,1I<,1B<,
1^<,4Z<,5?<,1^<,4D<,5W=,1P<,4N<,5`<,1P<
,4I<,5a=,1P<,1U<,1Y<,1W<,1P<,1W<,1Z<,4<
<,5Y=,1\<,4a<,5U<,1e<,4c<,5T<,1\<,4c<,5
P<,1f<,4U<,5e=
 1650 DATA1P<,4U<,5h<,1Y<,4W<,5`<,1P<,4
W<,5h<,1Z<,4a<,5Y=,1\<,4a<,5U=,1e<,1c<,
1\<,1c<,1f<,1e<,1\<,4Y<,5a<,1\<,4U<,5h<
,1Y<,4P<,5e<,1T<,4m<,5R=,1j<,4e<,5U<,1j
<,4a<,5Y<,1e<,4^<,5\<,1a<,4[<,5c<,1^<,4
W<,5g<,1[<,4R<
 1660 DATA5j<,1W<,4O<,5m<,1R<,4l<,5P<,1
h<,4c<,5T<,1h<,4`<,5W<,1c<,4\<,5[<,1`<,
4Y<,5a<,1\<,4U<,5e<,1Y<,4P<,5h<,1U<,4M<
,5l<,1P<,4j<,5O<,1g<,4d<,5R<,1g<,4`<,5T
<,1d<,4[<,5X<,1^<,4<<,5\<,1Y<,4U<,5h>,1
Y<,1R<,4U<,5e<
 1670 DATA4Y<,5a<,4\<,5e<,4[<,5^<,1W<,4
T<,5g>,1W<,1P<,4T<,5c<,4W<,5`<,4[<,5c<,
4Y<,5\<,1U<,4R<,5e>,1U<,1O<,4R<,5a<,4^<
,5U>,1a<,1[<,4h<,5T<,4g<,5U<,4e<,5R<,4d
<,5T=,1g<,4`<,5H<,1d<,4M<,5e=,1Y<,1T<,1
P<,1M<,1H<,1D<
 1680 DATA1H<,4<<,5A=,1h<,4k<,5M<,1h<,4
e<,5P<,1h<,4b<,5S<,1e<,4h<,5J=,1e<,1b<,
1e<,1^<,1\<,1Z<,1Y<,4<<,5W=,0<;,1f<,4j<
,5K<,1f<,4c<,5N<,1f<,4`<,5Q<,1c<,4f<,5H
=,1c<,1`<,1c<,1\<,1Z<,1Y<,1W<,4<<,5U=,1
e<,4h<,5I<,1e<
 1690 DATA4a<,5M<,1e<,4^<,5O<,1a<,4d<,5
F=,1a<,1^<,1a<,1[<,1Y<,1X<,1V<,5T=,4<<,
1c<,4f<,5H<,1c<,4`<,5K<,1c<,4]<,5N<,1`<
,4c<,5E=,1`<,1]<,1`<,1Y<,1W<,1U<,1T<,4<
<,5U=,1Y<,4^<,5R<,1a<,4`<,5Q<,1Y<,4`<,5
M<,1c<,4R<,5a=
 1700 DATA1M<,4R<,5^<,1U<,4T<,5]<,1M<,4
T<,5Y<,1W<,4^<,5U<,4a<,5Y<,4e<,5^<,4a<,
5Y<,4^<,5U<,4a<,5Y<,4[<,5R<,4^<,5U<,4a<
,5O<,4^<,5R<,4[<,5U<,4^<,5R<,4X<,5O<,4a
<,5R<,4`<,5L<,4^<,5O<,4]<,5M=,1`<,4c<,5
Q<,1`<,4]<,5T<
 1710 DATA1`<,4W<,5Q<,1Z<,4]<,5M<,1Z<,4
W<,5H<,1Z<,4T<,5E<,1Z<,4Y<,5A<,1W<,4U<,
5F=,1Y<,4^<,5I<,1Y<,4U<,5M<,1Y<,4R<,5I<
,1U<,4X<,5F=,1U<,4R<,5I=,1U<,4O<,5@<,1U
<,1T<,1R<,4<<,5Q=,1T<,4Q<,5`<,1M<,4K<,5
]<,1T<,4Q<,5Y<
 1720 DATA1K<,4<<,5I<,1Y<,4^<,5M<,1a<,4
`<,5E<,1Y<,4`<,5M<,1c<,4a<,5F<,1^<,4a<,
5O<,1e<,4c<,5H<,1`<,4c<,5Q<,1f<,4e<,5I<
,1a<,4e<,5R<,1h<,4f<,5K<,1e<,4c<,5S<,1a
<,4`<,5Q<,1a<,4c<,5N<,1e<,4f<,5K<,1c<,4
i<,5H<,1c<,4l<
 1730 DATA5E<,1c<,4a<,5F<,1j<,4f<,5?<,1
c<,4`<,5A<,1c<,4]<,5B=,1`<,4a<,5@=,1^<,
4Y<,5A=,1^<,4`<,5M=,1]<,4^<,5FC,1Y<,1U<
,1Y<,1R?
 1740 :
 1750 ON ERROR OFF
 1760 MODE7:IF ERR=17 END
 1770 REPORT:PRINT "at line";ERL
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

FUNCTION KEY LISTING OF ENVELOPES - T.J.L.Young

This will set a function key to display the parameters of any desired envelope (but will not work with O.S. 0.1). These are listed in the same order as in the ENVELOPE instruction, but the envelope number is supplied in response to the prompt:

```
*K.0MO.7:INP."Number of envelope";N%:@%=0:P."Envelope ";N%;:F.t%=0 TO 12:P.",";:
E%=?(&8C0+(N%-1)*16+t%):IF(t%>0 AND t%<4) OR (t%>6 AND t%<11) THEN IF E%>127 E%=
E%-1:T%=E%EOR&FF:P."-";:P.T%;:N.:P.:@%=10:EL.T%=E%:P.T%;:N.:P.:@%=10|M
```

QUICK SCREEN FILL

If you want to quickly define the contents of the text screen try poking to location &358, which is used to hold the character for blanking it out after a CLS instruction. For example

```
?&358=ASC("*"):CLS
```

in Mode 7 will produce a display full of asterisks. This is true for any text window that has been defined.

UNDOCUMENTED OSWORD CALLS IN ACORN DFS - A.Benham

In addition to OSWORD &7F (general disc read/write operations), the Acorn DFS also acts on OSWORDs &7D and &7E. The first of these two returns the number of disc write accesses, returned in the first byte of the parameter block (pointed to on entry of the OSWORD routine by the X and Y registers). OSWORD &7E is potentially more useful, returning the number of sectors expected on the disc.

The first and fourth byte of the parameter block are overwritten with &00. The second byte contains the lower eight bits of the 10 bit number specifying the number of sectors on the disc and the third byte contains the other 2 bits of the same 10 bit number. Neither call would appear to need parameters.

THE ACORN 6502 SECOND PROCESSOR

Reviewed by David Fell

When they launched their 65Ø2 Second Processor in March this year, Acorn described it as "the most important add-on for the Beeb so far...". In this in-depth technical review, David Fell explains why the fuss currently being made over the 65Ø2 Second Processor may be justified, and some of the snags that may catch the unwary user.

INTRODUCTION

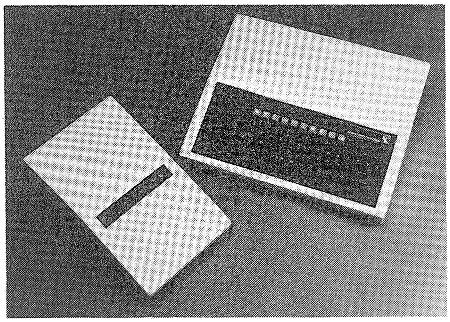
One of the major criticisms of the Beeb often levelled at Acorn is that there is not sufficient memory. For example, when using a disc-based Beeb in one of the high resolution modes, there is only about 6k of memory left into which a Basic program and its data must fit. If all of the user defined characters are required, and some extra memory is needed by other paged ROMs, then fitting a program into the available memory becomes very difficult indeed.

One solution, for £199.ØØ (inc. Vat), is the Acorn 65Ø2 Second Processor. This, and the associated software that comes with it, allows the user to run a Basic program that uses any mode, and still have 44k for his own program and data. Machine code programs have a massive 61k available!

65Ø2 SECOND PROCESSOR

The standard Beeb is referred to as the I/O processor (or Input/Output processor) when it has a second processor connected to it. This is because any input or output, to any peripheral that can be attached to a Beeb, will be performed by the I/O processor, and the processing of the language, or whatever application, will be performed by the second processor.

The Acorn 65Ø2 Second Processor comes in a box that matches the colour, design and depth of the Beeb, but is only half of the width (this being the new standard for most of Acorn's additions to the Beeb). The box contains the 65Ø2 Second Processor, 64k of RAM, interface to the Tube and its own power supply. There is a single switch at the back, a small fuse holder, and an adequate length of mains cable. When connected, the 65Ø2 Second Processor sits to the right of the Beeb



with a short ribbon cable connecting it to the Tube port, the rightmost of the plethora of ports underneath the Beeb.

When not required, the second processor can simply be switched off, and on re-initialisation the system reverts to a standard Beeb. The 65Ø2 Second Processor is supplied with two 16k EPROMs. One of these is essential for the running of the high speed Tube interface, which links the two processors together.

The DNFS EPROM that is supplied with the 65Ø2 Second Processor contains the Tube data-transfer routines and the latest versions of Acorn's disc and network filing systems (hence the name), and replaces any DFS or NFS ROMs that you may have in your machine. Since DFS Ø.9Ø (the only officially released Acorn DFS), Acorn have produced a large number of internal versions, and these have finally culminated in DFS 1.2Ø, which is the version contained in the DNFS EPROM. The NFS is version 3.34 (!), and incorporates a few small improvements over version 3.3Ø (says the manual).

The other EPROM is a new version of Basic that provides another 14k for your Basic programs (compared to the memory available when running either

Basic I or II in the 6502 Second Processor). With this new version of Basic, called 'Hi-Basic', a total of 44k of memory is now available for Basic programs and data irrespective of the screen mode in use. Compared with Basic II, this new version of Basic has also had a small amount of recoding done to some of its internal routines (although the author stresses that these are transparent to the user). For example, it now recognises COLOR as well as COLOUR, thus making it more suitable for the American market (which could feature heavily in the continued success of the BBC Micro).

BRIEF TECHNICAL NOTES

The 6502 Second Processor contains 64k of RAM, but, as with the standard Beeb, some of its RAM is used by the Operating System. The upper 2k of memory (&F800 to &FFFF) has some machine code copied into it on any Break, and this handles the software aspects of the communications between the two processors from the second processor's side. The DNFS EPROM copies some code down to the region &400 to &6FF on Break, and this handles the communications from the I/O side. With Basic (or any language), the start of free memory (otherwise known as OSHWM) is &800, which is just above the 1k that is always reserved for language workspace (&400 to &7FF). This, then, leaves free memory from &800 to &F7FF; which is 60k! Hi-Basic is a version of Basic II re-assembled to reside 14k higher up in memory (from &B800 instead on &8000) leaving 44k for the user in Hi-Basic.

FASTER PROCESSING

Although the 6502 Second Processor contains the same type of CPU as the standard Beeb, it runs 50% faster; which theoretically means that a program will run 50% faster. The actual speed improvement varies considerably, and the timings are related to the type of work being carried out by the I/O and second processors. One simple test program, that was designed to draw circles efficiently over the Tube, showed a speed improvement of over twice that of the standard Beeb (1.82 seconds on the Beeb, and 0.78 seconds on a 6502 Second Processor system). In general any improvements will depend on how a program is organised. For example, if a program is going to output to the screen, it will run faster if this is done as the results are calculated, and not at the end of the entire set of calculations (i.e. interleave calculation with screen output).

DOES IT WORK OVER THE TUBE THOUGH?

A lot has been said upon the merits of making a piece of software Tube compatible, though there has, until recently, been very little incentive to pursue this ideal, and little information from Acorn to help. Much of the software already available runs perfectly well on a standard Beeb (most games for example), and there is little point in suppliers attempting to achieve Tube compatibility for those products. The availability of more memory and greater speed is likely to have most impact on more serious software, particularly ROM based applications such as word processing, spreadsheets, and databases.

THE TUBE AND YOUR ROMS

When trying out ROM based software with a 6502 Second Processor, it soon becomes apparent that many existing ROMs do not work over the Tube. Many ROMs could benefit from the extra memory and speed available using a 6502 Second Processor, and yet very few appear to work correctly. The majority of those that appear to work with no problems, not surprisingly, originate from Acorn and Acornsoft. Many other ROMs that we have tried do not work over the Tube, though this may be a result of Acorn not having released adequate specifications rather than the software writers producing poor software.

Some ROMs have been written with no thought of Tube compatibility (e.g. Wordwise). Such ROM software has often been written with direct memory access to gain extra processing speed. (Given a relocated version of Wordwise that was Tube compatible, there could be about 50k available for text!)

The following table shows the results for the ROMs we have tested. Note that those marked as only partly working were ROMs that managed to perform some of their functions correctly, but did not fully work. Watford DFS did not manage its *EDIT command, but the main routines all worked, so I have classified this as working.

ROM Name	Does it work Over the Tube?
View	Yes
ViewSheet	Yes
BCPL	Yes
Wordwise	No
GROM	Partly
Exmon	Partly
Toolkit	No
Ultracalc	Yes
Vasm	No
Disc Doctor	Partly
Amcom DFS	No
Watford DFS	Yes
Termi	No
Beebfont	No

COPYING ROMS

There is a chapter about copying ROMs in the Acorn 65Ø2 Second Processor manual in which Acorn suggest that it is easier to copy the bulk of your language ROMs to disc, and then use, say, *BASIC (or whatever filename was given when the language was saved) to load the language from disc. Once the language is saved on disc, there is no need to keep the ROM in your machine, and so a socket can be vacated. (Is this perhaps because Acorn officially don't use ROM expansion boards?) However, there is a warning about infringement of other people's copyright, saying that specific permission must be obtained if the copyright material is not owned by Acorn. Acorn's manual also says "Where this copyright is owned by Acorn, the programs may be copied from ROM in the circumstances described in Acorn's 'Terms and Conditions of Sale and Use of Software' but not otherwise." Accessing languages in this way allows you to avoid purchasing a ROM board if you plan to use a 65Ø2 Second Processor.

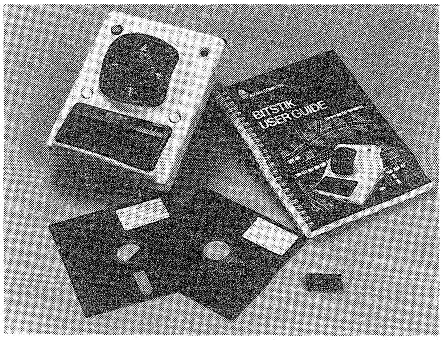
USING OTHER DISC FILING SYSTEMS

Since Acorn launched their DFS, a number of alternatives have appeared on the market. Some of these provided facilities for more files per directory, while others support double density disc drives. Having created lots of discs in special formats, it would be nice to be able to use them with a 65Ø2 Second Processor. Unfortunately, there is only one alternative DFS that appears to work over the Tube, and that is the Watford Electronics DFS. The main reasons for using the Watford DFS are that it allows you to create discs in a format that allow 62 files per directory, and it has a built in formatter and disc verifier. I used my WDFS while evaluating the 65Ø2 Second Processor, and encountered no problems, even when using Bitstik.

If you are going to use the WDFS, you need the DNFS EPROM installed as well, because the Tube transfer routines are still necessary. You can select either system by pressing Break and "D" for the Acorn DFS, or "W" for Watford DFS.

BITSTIK - THE 65Ø2 IN ACTION

Bitstik is a powerful graphics design package from Acornsoft for the 65Ø2 Second Processor user. It allows for the design, rapid storage and easy manipulation of complex pictures on the Beeb, with screen dumps to the Acorn JP1Ø1 Spark Jet printer. In order to fully realise the power of Bitstik, a graphics plotter is very desirable, and Acorn have a suitable package planned for release later on. The bulk of the software comes on an 8Ø track disc, but there is an 8k EPROM that controls most of the screen actions by directly accessing the screen memory. One point to note here is that if you place any code into a ROM, and call it via a "service call", then it can be guaranteed that execution will take



place in the I/O processor, and thus it is quite possible to access screen memory directly.

[From what we saw, Bitstik is a very fast system, and features some powerful options. We will be carrying an in-depth review of this exciting product in a future issue. Ed.]

TECHNICAL NOTES

The 6502 Second Processor is built around 64k of dynamic RAM, the Tube ULA, some buffer and refresh circuitry, and a 65C02B CPU. This is a 6502 fabricated in CMOS that runs at 3MHz. There is a 4k bootstrap EPROM, (but only the upper 2k contains anything meaningful) and the Tube is memory mapped within the region &FEF8 to &FEFF. There is a new * command provided by the Tube; the *GO command. This command takes a hex address, and executes a JSR to the address given. This can be used to enter the NLE (No Language Environment) through *GO F800. The last language can be re-entered through *GO 8000, for a normal language, and *GO B800 for Hi-Basic (assuming that it has not been corrupted in the meantime).

Once the operating system has taken out its 2k of memory from the very top of the memory map (&F800 to &FFFF), a small amount of operating system workspace must also be deducted, including zero page &EE to &FF. The Escape flag in the second processor will be kept in synchronisation with the one in the I/O processor, provided that interrupts are active, and the user can test for an Escape pending by BIT &FF, BMI errtrap, as was possible in the standard I/O processor. Page one is, as on all 6502 based systems, the stack, and the lower part of page two holds the operating system indirection vectors. Only the major vectors are supported, and some minor ones (e.g. OSRDRM) generate the error "Bad". Events can be active in both machines, but no OS calls are possible when running over the Tube. The rest of page two is an input buffer for the NLE, and also acts as a dump for error messages from the I/O processor. Page three does not appear to be used. Machine code users can use pages 4 through 7, as there is no language to claim this area. "Free" zero page in the I/O processor is used significantly by the Tube routines, so service calls should not stomp around there.

WRITING TUBE COMPATIBLE PROGRAMS

When writing software to work over the Tube, there are a few fundamentals to follow, and I'll now outline these.

The basis of writing Tube compatible software is to ensure that there is no direct access to any memory in the I/O processor (unless performed by a piece of code that is guaranteed to run in the I/O processor). For example, Wordwise (and some of the earlier Teletext adaptors) poke straight to the Mode 7 screen, and thus, when run in the second processor, nothing appears on the screen. Direct access to operating system variables should be avoided, as should direct reading and writing of the memory mapped areas. There are sufficient calls to read and write any necessary operating system variables, and even OSBYTE calls to read and write the memory mapped areas!

It should also be borne in mind that any of the soft key definitions, the soft character definitions, sound queues and disc/tape work space are always within the I/O processor, and these should thus not be accessed directly. Some people even use the addresses within the operating system of routines such as OSBYTE and OSWRCH instead of vectoring. This is also not Tube compatible, as it assumes an address that is only valid when running in the I/O processor.

These guidelines are, in general, equally applicable to Basic, machine code, and programs written in most languages, e.g. BCPL. Finally, please bear in mind that there is no point in attempting to make a program Tube compatible if there is no potential gain since non-Tube-compatible software can be run on a Tube based BBC micro system simply by switching off the second processor. Virtually no commercial games have been written in a Tube compatible manner, but then they function perfectly well without a second processor.

BEEBUG Workshop

SOME FURTHER STRING HANDLING ROUTINES

by Surac

Last month, the workshop dealt with some simple string handling techniques. Now, using some excellent routines sent in by Alan Dickinson, we will look at some further string functions in this very important area of programming in Basic.

String handling (the processing of characters) is probably the most important and useful activity that many micros are used for (for example in word processing). Because this is such a rich area, we have devoted a second workshop to this subject. This time we shall be looking at four functions; two to provide left and right justification of strings, and two useful additions to the RIGHT, LEFT and MID functions that are part of Basic. This month's functions are all used in a similar way; that is:

```
result$=FNaction(param1,param2,param3)
```

The type and number of the paramters will depend on the function itself.

To illustrate the usefulness of the first two functions imagine a program to input a name, and print it at a fixed postion on the screen. For the sake of screen format, you may require that the name will always occupy exactly, say, ten characters. What is needed is a function that can justify a string to a given length with a specific character, usually a space as here (but altering the space in quotes at line 1020 to "x" would allow the function to pad out with character "x"). When justifying like this in real life, there is often a need to left justify text (for example when printing name and address labels), and right justify numbers (especially money sums). For this reason both left and right justify functions are provided. Any numbers must be converted to string form using the STR$ function (see page 358 of the User Guide) before attempting to justify them.

LEFT AND RIGHT JUSTIFICATION FUNCTIONS

```
1000 DEF FNleft(A$,N%)
1010 IF LEN(A$)>N% THEN =A$
1020 =A$+STRING$(N%-LEN(A$)," ")

1100 DEF FNright(A$,N%)
1110 IF LEN(A$)>N% THEN =A$
1120 =STRING$(N%-LEN(A$)," ")+A$
```

The functions are called in the following manner:

```
   A$=FNleft(A$,10)
```

This would take the string A$, and pad it out to a string of 10 characters, with spaces added to the right hand end and, in this case, replace the old value of A$.

If the string supplied is longer than the length requested, then the routine exits without altering it. If you always want the function to deliberately truncate the string to the specified length, then you could add the following line:

```
      IF LEN(A$)>N% THEN =LEFT$(A$,N%)
```

as the second line in each case.

To describe how the two routines work, consider what needs to be done in order to pad a string of four characters up to a string of ten characters. The number of characters required to pad out the given string is the difference between the string length and the required length. In our example therefore, six spaces will need to be appended to either the left or the right of the original string depending on the function.

»

DELETING AND INSERTING CHARACTERS

The next two functions that we'll look at provide a couple of powerful and useful string handling functions. One of them allows you to remove a substring from within a larger string, with the remaining two parts closed up together. The other performs the reverse of this; i.e. it enables one string to be inserted at a given position within another.

STRING DELETION FUNCTION

```
1200 DEF FNdelete(A$,P%,L%)
1210 LOCAL A%
1220 IF L%<1 =A$
1230 IF P%<1 P%=1
1240 A%=LEN(A$)
1250 IF A%<P% =A$
1260 IF A%<P%+L% =LEFT$(A$,P%-1)
1270 =LEFT$(A$,P%-1)+MID$(A$,P%+L%)
```

STRING INSERTION FUNCTION

```
1300 DEF FNinsert(A$,P%,I$)
1310 LOCAL A%
1320 IF P%<1 P%=1
1330 A%=LEN(A$)
1340 IF A%<P% =A$+STRING$(P%-A%-1," ")
+I$
1350 =LEFT$(A$,P%-1)+I$+MID$(A$,P%)
```

The delete function is used to remove a specified group of characters from a larger string, and then close up the gap. Three parameters are required: the original string, the position from which the first character is to be deleted, and the number of characters to delete. For example:

A$=FNdelete(A$,3,5)
would take the string A$, and delete a a 5 character string, starting at position 3. If A$="ABCDEFGHIJKL", then the above operation would give "ABHIJKL". This is the type of action that takes place in word processing when a letter or word is deleted.

The workings of the routine are as follows. Lines 1210 and 1240 perform some brief initialisation and lines 1220 and 1230 carry out two initial checks for sensible parameter values. Having established that the parameters are valid, the routine needs to determines whether the deletion operation will remove the entire section after the starting character, or merely a section in the middle of the string. Line 1250 checks to see if an attempt is made to delete characters past the end of the string, and exits with the original string untouched if this is the case. Line 1260 then examines the effect of the deletion required, and returns the string to the left of the start postion if the deletion would leave no further characters to the right of those removed. If the routine has not already terminated, then it means that a proper substring is to be deleted, and the righthand section concatenated to the lefthand portion. Line 1270 performs this action and then exits.

The second of the two functions performs roughly the reverse of the above function: it allows for a string to be inserted into the middle of another string at a specified point (for example inserting a phrase or paragraph into existing text). For example, if A$="123ABC", then
A$=FNinsert(A$,4,"HI")
would give "123HIABC". The workings of the insert function are similar in some aspects to those of the delete function. There are two possible operations that the insert function will perform, and these are dependent upon whether or not the point at which the substring is to be inserted is actually within the main string, or beyond the end of it. Line 1340 caters for the second of these two possibilties, and returns a string consisting of the original string, a group of spaces to extend the original string to the necessary length, and then the substring. Line 1350 returns with the lefthand portion of the main string, the substring and then the remaining part of the main string, appended together in that order.

The routines presented in these workshops are designed for inclusion in your own programs when needed. If you have any small routines of your own, then send them into Surac at the address below. Next month we will be looking at another topic.

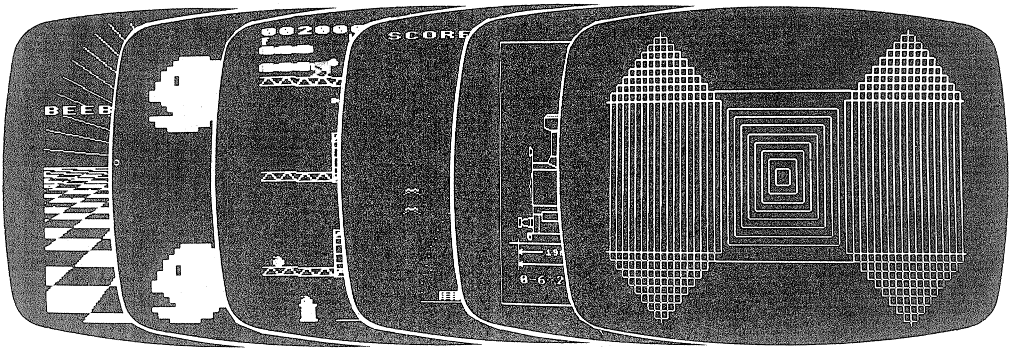
SURAC, BEEBUG, P.O. Box 50
St. Albans, Herts.

Tested on Basic I & II and O.S. 1·2

MULTI-SCREEN SLIDE SHOW (DFS)

by Tim Powys-Lybbe

Many programs available nowadays, such as ASTAAD in the March issue of BEEBUG, and packages such as Design and Teletext Editor from BEEBUGSOFT, allow screens to be designed and saved on disc or cassette. This Slide Show program is a disc utility which enables you to present a 'slide show' of screen designs, and provides very flexible control over the order of presentation.



The Slide Show program was produced with four main objectives:

1. To give a computer based slide show using 'slides' that were dumps of screen memory in any mode.

2. To provide a fixed order of slides that could be stepped through either singly or by jumping forwards or backwards.

3. To give an automatic show that could be left to run unattended.

4. To work for screens in all modes, Ø to 7.

The program allows any slide show to comprise up to 99 slides, which is both long enough for any show and more than will fit even on four sides of 8Ø track discs, except in Mode 7. The screens to be used in a slide show must be saved (*SAVE) as screen dumps on disc, with the load address appropriate to the screen mode in use (i.e. &3ØØØ for modes Ø, 1 and 2, &4ØØØ for mode 3, &58ØØ for modes 4 and 5, &6ØØØ for mode 6 and &7CØØ for mode 7).

The program works by creating a file which contains the order and names of the 'slides' to be used and the colours for each slide. The colour changing facility gives the impression, in a two colour mode, of multiple colour capability, adding to the appeal of your show. The slide show file can be edited to change anything, including deletion or insertion of more slides. The program will then run the slide show using this file to control the order of presentation. You can change this by single key controls to show the slides in any order.

The program starts with an initial menu of three choices to Make, Edit or Run a slide show. Whichever of these three main choices you make, you will also be asked for the disc drive on which the slide show file is to be saved or is to be found, and the name of the slide show file.

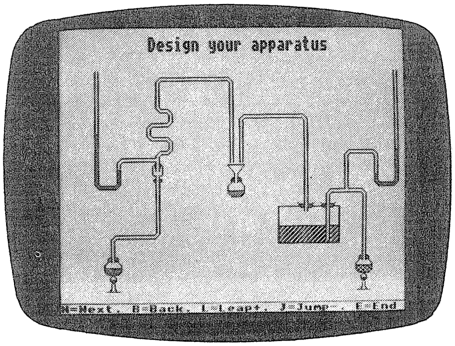
The slide show file will automatically be saved in disc directory R (for Run). This avoids any possible conflict between the name of a slide show file, and the name of a 'slide' in that file.

»

The Escape key is used to exit from any of the Make, Edit or Run facilities and to return to the start of the program. The Escape key will not crash the program or the slide show file, and in fact the slide show file can only be terminated by this means. Those parts of the program that are vulnerable to the use of the Escape key are protected by having the Escape key disabled.

RUNNING THE SLIDE SHOW

Before the slide show starts, you are asked for the time interval between slides. If you want an automatic show, enter a value of around 3Ø seconds - the BBC's CEEFAX pages change every 25 seconds, which is long enough for most people to read them. If you want total manual control enter a large figure, say 9999 seconds giving nearly 3 hours. Intervals of 6Ø seconds and upwards give a one line menu, displayed at the bottom of the screen, to show the controls available for stepping through the show. These controls are



available with shorter intervals, but the menu is not shown as it is less likely to be needed for automatic shows. On modes 2 and 5, the menu has to be highly abbreviated to fit into the 2Ø character width, so you are advised to learn the various controls in advance.

All the user controls are single key entries as follows:

B - Back to previous slide.
N - Forward to next slide.
L - Leap forward a specified number of slides.
J - Jump back a specified number of slides.
E - End slide show. The program asks for confirmation.

When you reach the last slide the next one will take you back to the start.

MAKING A SLIDE SHOW FILE

When you are creating a slide show file, the relevant details of each slide are displayed in column format on the screen. The eight items required in order, are:

1. Disc Drive Number. This is the disc drive on which this saved screen is to be found. Different slides can be on different drives so that those with twin double sided drives can use all their disc memory for long slide shows without changing discs.

2. Directory Letter. This is the directory letter for the slide's file name; it can be $ (the normal default), a numeral or a letter but other characters are inhibited.

3. Slide Name. This is the slide's file name which can be any combination of numerals and letters only. The same slide can be used as many times as you like, perhaps in different colours.

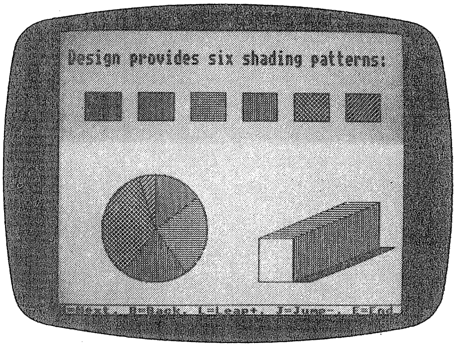
4. The screen mode in which the slide was saved. Notice that each slide can be in a different mode.

5 to 8. Colours. These are the BBC's colour numbers as given in the User Guide. Any number from Ø to 99 can be entered and will be deciphered on a base of 16 as usual. The first number is the background colour. Two-colour modes will only allow entry of two numbers and Mode 7 of course will not allow any colour numbers as colours are determined by the screen contents. The availability of all 16 colour numbers means that you can have flashing slides...

All entries must be followed by pressing the Return key. Do not worry if you make a mistake as you can edit the entries later.

You exit from making up the slide show by pressing the Escape key. To prevent any damage to the random access file, Escape is disabled during each slide entry. You are allowed two seconds at the end of each slide to press Escape. If you miss it, then repeat the last entry and press Escape immediately you get to the end of the line. Delete the extra entry with the editing facility later if necessary.

Pressing Break can crash the slide show file. You will not need to use any key near Break so accidents are unlikely.


Design provides six shading patterns:
N=Next, B=Back, L=Leap+, J=Jump-, E=End

EDITING A SLIDE SHOW FILE

Full editing facilities are provided. On entry the screen shows the entries for the first 18 slides together with a one line menu at the bottom of the screen. All entries to the editor must be followed by pressing the Return key. The menu allows:

1. Escape to exit from the editor back to the start of the program. This can be done at any time and all your amendments will be stored on the slide show file.

2. P (Print). This provides a printed copy on any 80 or more column printer that is on line. Obviously you need to have the *FX5, *FX6 and other commands set up suitably for your printer. Escape will escape from any problem.

3. S (Step). This steps the display to the next 18 slides. If you have reached the end of your file then it starts again at the beginning.

4. Line number. This refers to the number to the left of each line of slide data. To get into the editing facility proper, enter the line number concerned. The line number, by the way, is not stored in the file.

If a line number is entered, then a new one line menu is displayed. This menu repeats the chosen line number at the left and offers three choices (you can also Escape back to the beginning of the program):

E (Edit). This takes the cursor to each successive item of data in turn on the chosen line and deletes that item. You can either enter whatever you want or enter @ which will retain the previous entry. When you have finished editing that line, you can return to the previous menu.

D (Delete). This deletes the line number chosen and closes up the file so that it is one slide shorter.

I (Insert). This adds a duplicate entry on the line chosen, moves the original entry down one line, continuing this through the file so that it is one slide longer. You can then edit the new line to put any slide you want there. The reason for duplicating the existing entry is to ensure that the file has valid entries everywhere and cannot crash.

ENTERING THE PROGRAM INTO THE MICRO

Some emphasis has been given to ensuring that both the program and the slide show files will not crash. This means that the program has to include several routines to ensure that nothing can go wrong. Check the program carefully as you type it in, and save it on disc.

```
 10 REM PROGRAM SLIDES
 20 REM AUTHOR  Tim Powys-Lybbe
 30 REM VERSION B1.1
 40 REM BEEBUG  JUNE 84
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
100 MODE0
110 ON ERROR PROCError:MODE7:END
```

➜

```
 12Ø PROCInitialise
 13Ø PROCSlideShow
 14Ø PROCRun
 145 END
 15Ø :
1ØØØ DEF PROCError
1Ø1Ø ON ERROR OFF:*FX3
1Ø2Ø VDU3:PROCOpenCommand:PROCReset:*F
X4
1Ø3Ø IFERR=17 THENPRINTCR$"End/Start?
(e/s) ";:A$="ES":PROCInput(1) ELSEREPOR
T:PRINT" at ";ERL;:PROCCursorOn:END
1Ø4Ø IFK$="S" THEN RUN
1Ø5Ø ENDPROC
1Ø6Ø :
1Ø7Ø DEF PROCInitialise
1Ø8Ø PROCMode7Colours:PROCReset:VDU22,
7:PROCNoCursor:*FX4,1
1Ø9Ø DIMQ%4Ø:REM Space for operating s
ystem calls
11ØØ DIMC%(3): REM First four colours
111Ø L%=35: REM Length of file entry f
or each slide
112Ø ENDPROC
113Ø :
114Ø DEF PROCMode7Colours
115Ø CR$=CHR$129:CG$=CHR$13Ø:CY$=CHR$1
31:CZ$=CHR$156:CW$=CHR$157
116Ø ENDPROC
117Ø :
118Ø DEF PROCReset
119Ø CLOSE#Ø:m%=7:Z%=Ø:E$="Ø123456789"
:F$="ABCDEFGHIJKLMNOPQRSTUVWXYZ":PROCES
Con
12ØØ ENDPROC
121Ø :
122Ø DEF PROCSlideShow
123Ø PROCHeading:PRINT''CG$"Do you wan
t to"''CG$"  1"CR$"Run"CG$"a slide show
"'CG$"  2"CR$"Make"CG$"a slide show"'CG
$"  3"CR$"Edit"CG$"a slide show"''CG$"E
nter your choice (1 to 3):"CR$;
124Ø A$="123":PROCInput(1):B%=C
125Ø PRINT''CG$"What"CR$"disc drive"CG
$"do you want to use"'CG$"for the slide
 show file (Ø to 3)?"CR$;
126Ø A$="Ø123":PROCInput(1):D%=C
127Ø Y%=VPOS:REPEATPRINTTAB(Ø,Y%+2)CG$
"What is the"CR$"name"CG$"of the slide"
'CG$"show file?"CR$;
128Ø A$=E$+F$:PROCInput(7):B$=":"+STR$
D%+".R."+C$:IFB%=2 THENPROCOS("SAVE "+B
$+" 8ØØØ+1ØØØ")
129Ø A=OPENUP(B$):IFA=Ø THENPROCOpenCo
mmand:PRINTCR$"File not found.  Try aga
in.";:VDU7,26
13ØØ UNTILA:PROCOpenCommand:VDU26
131Ø IFB%=1 THENPRINTTAB(Ø,Y%+5)CG$"Wh
at is the"CR$"maximum interval"CG$"betw
een"'CG$"slides in seconds (9999 for full"
'CG$"manual control)?"CR$;:A$=E$:PRO
CInput(4):T%=C:ENDPROC
132Ø IFB%=2 THENPROCMake ELSEPROCEdit
133Ø :
134Ø DEF PROCMake
135Ø CLS:PROCHeading:PRINT'CR$"ESC exi
ts for 2 secs at line end only"
136Ø PROCHead:K%=Ø:PRINT#A,K%:REPEATK%
=K%+1:PROCData(K%):PROCEnd
137Ø C=INKEY(2ØØ):PRINT:UNTILFALSE
138Ø :
139Ø DEF PROCEdit
14ØØ E$=E$+"@":INPUT#A,K%:REPEATI%=Ø:P
TR#A=5:REPEATVDU26,12:PROCHeading:PROCH
ead:J%=I%:REPEATI%=I%+1:Y%=7+I%-J%:PROC
PrintSlide:UNTILI%=K% ORI%=J%+1Ø:A$="re
starts":IFI%<K% A$="continues"
141Ø IFK%>1Ø PRINTTAB(8,21)CR$"S=Step
"A$" list."
142Ø REPEAT:REPEAT:PROCOpenCommand:PRI
NTCR$"ESC=Exit P=Print S=Step OR line n
o: ";:A$=E$+"SP"
143Ø PROCInput(2):UNTILK$="S" OR(C>J%
ANDC<=I%) ORK$="P":IFK$="S" THEN UNTIL
TRUE:UNTILI%=K%:UNTILFALSE
144Ø IFK$="P" THENPROCPrint:UNTILTRUE:
UNTILTRUE:UNTILFALSE
145Ø T%=C:PTR#A=(T%-1)*L%+5:PRINT'CR$"L
ine ";T%":  E=Edit D=Delete I=Insert ";
146Ø A$="EDI":PROCInput(2)
147Ø IFK$="E" THEN PRINT'CHR$136 TAB(1
1)CR$"@=Copy old entry";:VDU26,31,Ø,T%-
J%+7:PROCData(T%):PTR#A=5+I%*L%:UNTILFA
LSE
148Ø IFK$="I" THENPROCMore ELSEIFK$="D
" THENPROCLess
149Ø UNTIL TRUE:UNTIL TRUE:UNTIL FALSE
15ØØ :
151Ø DEF PROCHead
152Ø PRINT'CR$SPC(23)" Colours: (Ø-15)
"CR$" Drive Dir  Slide  Scrn Back  [ Fo
re ]"'CR$"   No."SPC(7)"Name  Mode  Ø
 1   2   3"'
153Ø ENDPROC
154Ø :
155Ø DEF PROCData(I%)
156Ø Y=VPOS:PRINTTAB(Ø,Y)CG$TAB(3-LENS
TR$I%,Y);I%;
157Ø RESTORE165Ø:PROCESCoff:FORJ=ØTO7:R
EADA%,B%:D$=STRING$(B%," "):A$=E$:IFJ=1
ORJ=2 THENA$=A$+F$:IFJ=1 THENA$=A$+"$"
158Ø REPEATPRINTTAB(A%,Y)D$TAB(A%,Y);:
PROCInput(B%):UNTILNOT(J=Ø ANDC>3) ANDN
OT(J=3 ANDC>7)
159Ø IFC$="@" THENINPUT#A,C$:PRINTTAB(
A%,Y)C$; ELSEPROCPrintRealData
16ØØ IFJ=3 THENm%=VALC$:PROCNoOfColours
161Ø IFm%=7 ANDJ>2 THEND$="  ":PRINT#A
,D$,D$,D$,D$:J=7
```

```
 162Ø IFM%=1 ANDJ>4 THENPRINT#A,D$,D$:J
=7
 163Ø NEXT:PROCESCon
 164Ø ENDPROC
 165Ø DATA5,2,9,2,12,7,21,2,25,2,29,2,3
3,2,37,2
 166Ø :
 167Ø DEF PROCMore
 168Ø K%=K%+1:FORI%=1TOK%-T%:PTR#A=5+(K
%-1-I%)*L%:PROCMove(L%):NEXT:PROCEnd
 169Ø ENDPROC
 17ØØ :
 171Ø DEF PROCLess
 172Ø IFK%<2 THENENDPROC ELSEIFT%<K% TH
ENPTR#A=PTR#A+L%:REPEATPROCMove(-L%):UN
TILFNEnd
 173Ø K%=K%-1:PROCEnd
 174Ø ENDPROC
 175Ø :
 176Ø DEF PROCEnd
 177Ø A%=PTR#A:PROCESCoff:PTR#A=Ø:PRINT
#A,K%:PROCESCon:PTR#A=A%
 178Ø ENDPROC
 179Ø :
 18ØØ DEFFNEnd
 181Ø IFPTR#A=5+K%*L% THEN=1 ELSE=Ø
 182Ø :
 183Ø DEF PROCMove(C%)
 184Ø PROCESCoff:FORI=ØTO7:A%=PTR#A:INP
UT#A,A$:B%=PTR#A:PTR#A=A%+C%:PRINT#A,A$
:PTR#A=B%:NEXT:PROCESCon
 185Ø ENDPROC
 186Ø :
 187Ø DEF PROCOpenCommand
 188Ø MOVEØ,36:DRAW128Ø,36:l%=39:IFZ% A
ND(m%=2ORm%=5) THENl%=19
 189Ø n%=31:IFZ%=Ø ORINSTR("367",STR$m%
)>Ø THENn%=24
 19ØØ VDU28,Ø,n%,l%,n%,12
 191Ø ENDPROC
 192Ø :
 193Ø DEF PROCInput(length)
 194Ø LOCALX,Y:X=POS:Y=VPOS:C$="":PRINT
TAB(X,Y)STRING$(length," ")STRING$(leng
th,CHR$8);:PROCCursorOn:REPEATB$=GET$
 195Ø IFINSTR(A$,CHR$(ASCB$ AND&DF))>Ø
ORINSTR(A$,B$)>Ø THENC$=LEFT$(C$+B$,len
gth):PRINTTAB(X,Y)C$;
 196Ø IFB$=CHR$127 ANDLENC$>Ø THENC$=LE
FT$(C$,LENC$-1):VDU127
 197Ø UNTILLENC$>Ø AND(B$=CHR$13 ORleng
th=1):C=VALC$:K$=CHR$(ASCC$ AND&DF):PRO
CNoCursor
 198Ø ENDPROC
 199Ø :
 2ØØØ DEF PROCOS(A$)
 2Ø1Ø LOCALX%,Y%:X%=Q%:Y%=Q%DIV256:$Q%=
A$:CALL&FFF7
 2Ø2Ø ENDPROC
 2Ø3Ø :
 2Ø4Ø DEF PROCCursorOn
 2Ø5Ø VDU23;11,255;Ø;Ø;Ø;
 2Ø6Ø ENDPROC
 2Ø7Ø :
 2Ø8Ø DEF PROCNoCursor
 2Ø9Ø VDU23;11,Ø;Ø;Ø;Ø;
 21ØØ ENDPROC
 211Ø :
 212Ø DEF PROCColours
 213Ø IFm%<7 THENLOCALI%:FORI%=ØTOM%:VD
U19,I%,C%(I%);Ø;:NEXT
 214Ø ENDPROC
 215Ø :
 216Ø DEF PROCPrint
 217Ø *FX3,1Ø
 218Ø PRINT'" ","Drive Dir","Slide","Sc
reen",,"Colours"'" ","No","Name","Mode"
,"Ø","1","2","3"'
 219Ø PTR#A=5:FORI%=1TOK%:INPUT#A,A$:PR
INT;I%,A$"      ";:FORJ%=ØTO6:INPUT#A,A
$:PRINTA$,;:NEXT:PRINT:NEXT:VDU12:*FX3
 22ØØ ENDPROC
 221Ø :
 222Ø DEF PROCPrintSlide
 223Ø PRINTTAB(Ø,Y%)CG$TAB(3-LENSTR$I%,
Y%);I%;:RESTORE165Ø:FORI=ØTO7:READA%,B%
:INPUT#A,A$:PRINTTAB(A%,Y%)A$;:NEXT
 224Ø ENDPROC
 225Ø :
 226Ø DEF PROCPrintRealData
 227Ø IFJ>1 THENC$=LEFT$(C$+D$,B%) ELSE
C$=RIGHT$(C$,1)
 228Ø PRINT#A,C$
 229Ø ENDPROC
 23ØØ :
 231Ø DEF PROCESCoff
 232Ø *FX22Ø,2ØØ
 233Ø ENDPROC
 234Ø :
 235Ø DEF PROCESCon
 236Ø *FX22Ø,27
 237Ø ENDPROC
 238Ø :
 239Ø DEF PROCRun
 24ØØ Z%=1:PROCNoCursor:INPUT#A,K%:V%=(
K%-1)*L%+5:REPEATPTR#A=5:REPEATINPUT#A,
A$,B$,C$:D$="LOAD :"+A$+"."+B$+"."+C$
 241Ø INPUT#A,C$:m%=VALC$:VDU22,m%:PROC
NoCursor:PROCNoOfColours:FORI=ØTO3:INPU
T#A,A$:C%(I)=VALA$:NEXT:PROCColours:PRO
COS(D$):*FX15
 242Ø PROCKeys:UNTILFNEnd:UNTILFALSE
 243Ø :
 244Ø DEF PROCKeys
 245Ø REPEAT:IFT%<6Ø THENMOVEØ,36:PLOT7
,128Ø,36 ELSEPROCOpenCommand:PROCCursor
On:IFm%=2 ORm%=5 THENPRINT"Next/Back/L+
/J-/End"; ELSEPRINT"N=Next. B=Back. L=L
eap+. J=Jump-. E=End";
```

Continued on page 27

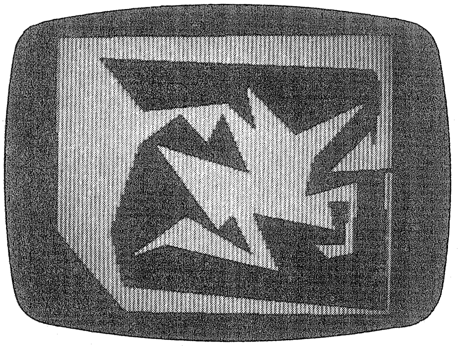
Tested on Basic I & II and O.S. 1.2

A FAST COLOUR FILL ROUTINE (16K)

by Stephen Todd

Here is an exceptionally useful utility that you could use in a whole variety of your programs. It is a routine that will fill any area on the screen in any colour and in any graphics mode very quickly indeed.

The ability to colour in any irregular area on the screen is a feature of most commercial graphics packages. O.S 1.2 does allow some crude fill routines as versions of the PLOT command (PLOT 77), but writing a comprehensive fill routine is still difficult, and in Basic the results are slow. The machine code routine presented here is one that you can readily incorporate in your own programs (whether they be in Basic or machine code), or load from disc or cassette as required. The routine is listed as part of a program FILL to demonstrate just how versatile it is.



To see this demonstration, type in the program, and after saving to cassette or disc, just run the program. The program draws a shape on the screen and then fills this with a randomly chosen colour, starting from a random position.

COMMENTS IN MACHINE CODE PROGRAMS.

In machine code, program comments are included by placing the text of the comment after a back-slash character '\'. When entering machine code programs all such comments can be omitted to save time.

The fill routine itself is contained in the procedure PROCassemble, from lines 1160 to 2260, and you may wish to save this separately using *SPOOL (see User Guide page 402) for inclusion in your own programs (using *EXEC). See this month's article in the 'Beginners Start Here' series for more information on *SPOOL and *EXEC (also in the User Guide page 402).

To use the routine, your program must first call the procedure PROCassemble to set up the machine code routine, as in the demonstration program at line 130. To fill an area you should select the colour using the GCOL command and use MOVE to select a suitable starting point within the area to be coloured (lines 160 and 180). The fill routine is then called by using the command:

```
CALL fill
```

As a simple example, try the following after you have correctly entered the FILL program:

```
DELETE 140,1150
140 GCOL 0,2
150 MOVE 300,300
160 DRAW 600,300:DRAW 600,600
170 DRAW 300,600:DRAW 300,300
180 MOVE 400,400:DRAW 500,400
190 DRAW 500,500:DRAW 400,500
200 DRAW 400,400:GCOL1,1:MOVE350,350
210 CALL fill:GCOL1,3:MOVE 450,450
220 CALL fill:END
```

The sequence of select colour (using GCOL), move to starting point (using MOVE), and then fill is used twice in lines 200 to 220.

There is another way to use this fill routine, and that is by assembling the machine code and then *SAVEing it on cassette or disc. Then when you need to use the routine, you can *LOAD it back into memory and call the code at the load address. To do this we need to delete line 117Ø and alter line 118Ø to P%=&2EØØ (or any suitable address below screen memory with 45Ø bytes to spare). For example, if we wanted to use the fill routine in mode 5, we could alter line 118Ø to P%=&56ØØ (mode 5 screen memory starts at &58ØØ leaving 512 bytes), and then assemble the code. Then we would save the code using:

```
   *SAVE "fill" 56ØØ +45Ø 56ØØ 56ØØ
```

When we want to use this fill routine, we could include the following lines in our program:

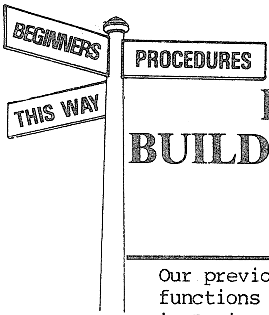
```
   13Ø HIMEM=&56ØØ
   14Ø *LOAD fill
   21Ø GCOL 1,2:MOVE X%,Y%:CALL &56ØØ
```

Changing the value of HIMEM to the same load address protects the fill routine from being overwritten by any data. The value of &56ØØ is appropriate for both modes 4 and 5, while &2EØØ is the corresponding figure for the other graphics modes (Ø, 1 and 2).

```
   1Ø REM PROGRAM FILL
   2Ø REM VERSION B1.Ø
   3Ø REM AUTHOR  S.TODD
   4Ø REM BEEBUG  JULY 1984
   5Ø REM PROGRAM SUBJECT TO COPYRIGHT
   6Ø :
  1ØØ ON ERROR GOTO 228Ø
  11Ø MODE2
  12Ø VDU23,1,Ø;Ø;Ø;Ø;
  13Ø PROCassemble
  14Ø GCOLØ,1
  15Ø PROCbox
  16Ø VDU19,1Ø26;Ø;Ø:GCOL RND(8)+51,2
  17Ø REPEAT:X%=RND(1279):Y%=RND(1Ø23):
UNTILPOINT(X%,Y%)=Ø
  18Ø MOVEX%,Y%
  181 CALL fill
  182 END
  19Ø :
 1ØØØ REM Demonstration shape
 1Ø1Ø DEF PROCbox
 1Ø2Ø MOVE1ØØ,25Ø
 1Ø3Ø RESTORE 1Ø7Ø
 1Ø4Ø READ x,y:IF x=-1 GOTO 114Ø
 1Ø5Ø DRAW x,y:GOTO1Ø4Ø
 1Ø6Ø
 1Ø7Ø DATA37Ø,37Ø,56Ø,19Ø,52Ø,35Ø,63Ø,26
Ø,94Ø,29Ø,11ØØ,1ØØ,9Ø,14Ø,6Ø,2ØØ,8Ø,4ØØ
 1Ø8Ø DATA9Ø,35Ø,25Ø,5ØØ,1Ø,49Ø,1Ø,1Ø,1
27Ø,2Ø,126Ø,7ØØ,1ØØØ,1ØØØ,Ø,1Ø23,Ø,5ØØ
 1Ø9Ø DATA4Ø,5ØØ,3Ø,1ØØØ,1ØØØ,9ØØ,1Ø5Ø,
7ØØ,95Ø,4ØØ,8ØØ,3ØØ,7ØØ,4ØØ,64Ø,3ØØ,55Ø
 11ØØ DATA5ØØ,9ØØ,4ØØ,66Ø,7ØØ,6ØØ,65Ø,5
3Ø,78Ø,78Ø,65Ø,1ØØØ,8ØØ,85Ø,75Ø,42Ø,85Ø
 111Ø DATA52Ø,64Ø,22Ø,7ØØ,24Ø,6ØØ,18Ø,6
ØØ,19Ø,75Ø,3ØØ,75Ø,25Ø,8ØØ,15Ø,78Ø,14Ø
 112Ø DATA53Ø,33Ø,58Ø,3ØØ,45Ø,1ØØ,25Ø
 113Ø DATA -1,-1
 114Ø ENDPROC
 115Ø :
 116Ø DEF PROCassemble
 117Ø DIM PT% 45Ø:FORT%=ØTO2 STEP2
 118Ø P%=PT%
 119Ø [OPT T%
 12ØØ .fill
 121Ø LDA #Ø:STA &76:LDA #12:STA &77:LD
A &76:SEC:SBC #1:STA &76
 122Ø BCS runit:DEC &77
 123Ø .runit
 124Ø LDA #Ø:STA &7E:LDA #&87 \ Make su
re we are in a graphics mode
 125Ø JSR &FFF4:TYA:TAX:LDA modetable,X
:BPL rightmode
 126Ø JMP wrongmode
 127Ø .rightmode
 128Ø STA &78
 129Ø LDY #Ø \ Get the last graphics po
int into &7Ø-&73
 13ØØ LDX #&88:LDA #13:JSR &FFF1:LDX #4
 131Ø .again
 132Ø LDA &8B,X:STA &6F,X:DEX:BNE again
 133Ø
 134Ø LDY #Ø:LDX #&7Ø:LDA #9:JSR &FFF1:
LDA &74:STA &7C
 135Ø BEQ repeat1 \ If its background g
o colour fill in current GCOL
 136Ø .error
 137Ø LDA #7:JSR &FFEE:CLC:RTS \ else r
eturn to the calling program
 138Ø .repeat1
 139Ø BIT &FF \ Test for ESCAPE key and
 e xit if pressed
 14ØØ BPL notescape
 141Ø LDA #&7E:JSR &FFF4:JMP error
 142Ø .notescape
 143Ø LDA #&FF \ Set draw/miss flags on
 144Ø STA &79:STA &7A
 145Ø LDA #25 \ Do PLOT77,xcoor,ycoor
 146Ø JSR &FFEE:LDA #77:JSR &FFEE:LDX #Ø
 147Ø .back
 148Ø LDA &7Ø,X:JSR &FFEE:INX:CPX #4:BN
E back
 149Ø LDY #Ø \ Get back the co-ordinate
s at the end of the PLOT77 line
 15ØØ LDX #&88:LDA #13:JSR &FFF1:LDA &8
C:STA &7Ø:LDA &8D:STA &71
```

»

```
 1510 LDA &72 \ Up one pixel in the Y d
irection
 1520 CLC:ADC #4:STA &72:BCC repeat2:IN
C &73
 1530 .repeat2
 1540 LDY #0 \ Get the colour of that p
oint
 1550 LDX #&70:LDA #9:JSR &FFF1:LDA &74
:BPL notpoint
 1560 JMP point \ If the point was off
screen
 1570 .notpoint
 1580 CMP &7C:BEQ check1 \ If its backg
round check the up draw flag
 1590 LDA #&FF   \ else set the up draw
 flag
 1600 STA &79:BNE nextx \ and try next
point along x axis
 1610 .check1
 1620 LDA &79 \ If up flag unset dont d
raw line
 1630 BEQ nextx
 1640 JSR storepoint \ else remember th
is point
 1650 LDA #0 \ and unset the draw flag
 1660 STA &79
 1670 .nextx
 1680 LDA &70 \ Work out the next x poi
nt and check if we have reached
 1690 SEC:SBC &78:STA &70 \ the end of
the line
 1700 BCS nodecrement:DEC &71
 1710 .nodecrement
 1720 LDA &88:SEC:SBC &70:STA &75:LDA &
89:SBC &71:BNE repeat2
 1730 LDA &75:CMP &78:BNE repeat2
 1740 LDA &70 \ Now look below the line
 and store any relevant points
 1750 CLC:ADC &78:STA &70:BCC 18:INC &71
 1760 .18
 1770 LDA &72:SEC:SBC #8:STA &72:BCS no
ty:DEC &73
 1780 .noty
 1790 LDY #0:LDX #&70:LDA #9:JSR &FFF1:
LDA &74:BMI point:CMP &7C
 1800 BEQ check2:LDA #&FF:STA &7A:BNE i
ncrex
 1810 .check2
 1820 LDA &7A:BEQ increx:JSR storepoint
:LDA #0:STA &7A
 1830 .increx
 1840 LDA &70:CLC:ADC &78:STA &70:BCC n
oin:INC &71
 1850 .noin
 1860 LDA &70:SEC:SBC &8C:STA &75:LDA &
71:SBC &8D:BNE noty
 1870 LDA &75:CMP &78:BNE noty
 1880 .point
 1890 JSR popstack \ Checked all of the
 last drawn line so now get
 1900 BCS endfill \ a new start point i
f any are left
 1910 JMP repeat1
 1920 .endfill
 1930 CLC:RTS \ leave this program
 1940 .popstack \ Pop the last item on
the stack adjusting the stack pointer
 1950 LDA &7E:BNE pointhere:SEC:RTS
 1960 .pointhere
 1970 SEC:SBC #4:STA &7E:LDA &76:SEC:SB
C #4:STA &76:BCS poppoint:DEC &77
 1980 .poppoint
 1990 LDY #4
 2000 .reppop
 2010 LDA (&76),Y:STA &6F,Y:DEY:BNE rep
pop:CLC:RTS
 2020 .storepoint \ Push a point on the
 stack if there is enough room
 2030 LDA &7E:CMP #&FC:BEQ cantstore:CL
C:ADC #4:STA &7E:LDY #4
 2040 .repstore
 2050 LDA &6F,Y:STA (&76),Y:DEY:BNE rep
store:LDA &76
 2060 CLC:ADC #4:STA &76:BCC returnstor
e:INC &77
 2070 .returnstore
 2080 RTS
 2090 .cantstore
 2100 JSR sound:RTS
 2110 .sound \ Issue the error 'beep'
 2120 LDA #7:LDX #sounddata MOD 256:LDY
 #sounddata DIV 256:JSR &FFF1:RTS
 2130
 2140 ]
 2150 :
 2160 modetable=P%:REM Step size in the
 x direction in all the modes
 2170 !P%=&FF080402:P%!4=&FFFF0804
 2180 :
 2190 sounddata=P%+8:REM Data for beep
 2200 P%!8=&FFF70011:P%!12=&100FA
 2210 :
 2220 wrongmode=P%+16:REM error message
 2230 P%?16=0:P%?17=&FF:$(P%+18)="Not g
raphics mode"+CHR$0
 2240 :
 2250 NEXT
 2260 ENDPROC
 2270 :
 2280 ON ERROR OFF
 2290 MODE7:IF ERR=17 END
 2300 REPORT:PRINT" at line ";ERL
```



BEGINNERS START HERE BUILDING A PROCEDURE/FUNCTION LIBRARY

by Peter Lewis

Our previous two articles for beginners have shown how useful functions and procedures are in both designing and writing programs in Basic. This month, Peter Lewis shows you how to put all your most useful functions and procedures together to form a personal library.

In the previous two articles in this series, I have described in some detail how to write your own functions and procedures, and how useful these language features are in helping the design and development of computer programs. If you look at the listings of many of the programs that you see published in BEEBUG, you will see how often the authors have made use of these features. Our new BEEBUG Workshop series also makes extensive use of functions and procedures to express many quite simple ideas.

Once you start writing your own functions and procedures, you will soon realise that many of them can be used not just in one program, but in many different programs. In fact, the shorter and more basic the procedure is the more frequently you are likely to find yourself dusting it off to use in yet another program. Even where exactly the same procedure is not appropriate, much programming time can often be saved by loading in an existing procedure and modifying it to suit, rather than programming from scratch.

In order to make the most of this idea, we need to have some way of saving all of our most useful procedures and functions, and of course, of adding those we select to any new program that we are developing.

Before proceeding further there is a technical point that needs to be explained. When you create a program by typing lines of Basic into the computer, the screen display always consists entirely of ASCII characters (see the appendix on page 486 of the User Guide) corresponding to the characters typed in from the keyboard. Thus the program line:

```
100 PRINT"fred"
```

consists of a string of 15 characters when displayed on the screen.

The same instruction is stored differently in the computer's memory to save space. Line numbers are coded and all Basic keywords are held in so called 'tokenised' form (see User Guide page 483). This has the advantage that each Basic keyword can be stored in a single byte of memory irrespective of the length of the keyword. As a result, the instruction given above would occupy only 12 bytes of memory.

Now you may well be wondering about the point of all this. Normally, programs are saved and loaded, using either cassette or disc, in a tokenised form. But the commands SAVE and LOAD only deal with whole programs. They do not allow you to load a program into memory and then append another program (function or procedure) to the program already loaded. This can only be done using the commands *SPOOL and *EXEC, and both of these use an ASCII, not tokenised format.

SETTING UP AND USING A LIBRARY

All your main programs should be saved and loaded in the usual way. Whenever you have a function or procedure which you think would be worth adding to your library then proceed as follows. First ensure that only the instructions for defining the function or procedure are left in memory (starting with DEF). Suppose we want to save this under the name PROCFN. Type:

```
*SPOOL PROCFN       <return>
LIST          <return>
*SPOOL        <return>
```

This will save an ASCII version of the function or procedure on either disc or cassette. If at a later date you want to add this to a new program which you are developing type in the following command, assuming that the new program so far developed is already in memory.

```
*EXEC PROCFN
```

The procedure or function PROCFN will then be added to the program already in memory. In effect, the command *EXEC enters the contents of the file PROCFN (or whatever you select) into memory just as though it was typed in from the keyboard.

SOME PRACTICAL POINTS

One important practical point concerns line numbers. When you save a function or procedure in your library, you will have no idea of the line numbers you will be using in future programs. Thus when you add a function or procedure to a program there may be an overlap of line numbers with the program already in memory, and some of the lines will be lost (just as they would be if you typed in a line from the keyboard that had the same line number as an existing line).

You can't help this, but at least if you are aware of the possibility you can check, and if necessary renumber the program in memory before adding the function or procedure. You will also find it helpful to number all your library routines from, say, line 10000 upwards. This will leave plenty of room in the earlier line numbers for the main program. It may also be a good idea to record the line number range for each routine.

Another good idea is to allocate one or more discs or cassettes for use as your function and procedure library. If you find that saving very short functions or procedures is wasteful, then you might like to save several together as a single file. This obviously makes most sense if the individual functions and procedures are related in some way, for example graphics routines or string handling routines. You may also choose to designate different discs or cassettes in the same thematic way. You will clearly need to keep a record in some form of where each function or procedure is stored and what it does.

Building a library of all your most useful functions and procedures is a good way of organising your programming, and one that many computer professionals use as a matter of course. As you develop your programming skills and become more sophisticated, you may well find the LINK utility published in BEEBUG Vol.2 No.7 and the Function/Procedure Overlay utility in BEEBUG Vol.3 No.2 add greatly to the value of your library, though to make the most of these you really need a disc system.

You may also decide that many of the functions and procedures you see published in BEEBUG are also worth including in your library, and our Workshop series should prove particularly useful in this respect.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

INSTANT VDU EFFECTS USING CONTROL - A.Porter

It's possible to select any screen mode, foreground or background colour without having to resort to lengthy VDU codes. Modes 0 to 7 are easily selected by typing Ctrl-V followed by the required mode number (though notice that HIMEM is not reset to match the mode chosen). In mode 1 a control code sequence of 'S06000' (equivalent to a VDU19 command) followed by 'S11000' will produce a cyan background with red text. (This has been hinted at by several of our contributors as being a very useful trick for adding colour to View text. This must be done whilst in command mode).

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

A PRINTER SPOOLER UTILITY (16K)

by D.S. Peckett

Printing is a time consuming activity that all too often can tie up your micro for a long period of time. The printer spooler described here is a most useful utility which will allow you to carry on working, and do your printing at the same time.

INTRODUCTION

It is a fact of computing life that a system is never quite fast enough. The Beeb is better than most micros but still, as soon as you start printing, your whole system is tied up while it generates a listing of your latest magnum opus. Wouldn't it be nice to be able to carry on using the computer while it was printing?

"Professional" computers (ie those costing several thousand pounds plus) often incorporate a "spooler", which is a system allowing the computer to print a file while it carries on with its normal processing. An alternative is to purchase a piece of hardware called a buffer for your printer which allows large amounts of text to be transferred fast to the printer which deals with this at its own speed leaving the computer free for other work. But this all costs money, up to £1ØØ in fact. The small buffer built into many printers now available will hold very little text (even a 2k buffer is not that large).

This article describes a short program which will add a software spooler to a BBC micro, allowing it to print text as a background task while simultaneously processing a totally different program as a foreground task. The routine works with both tape and disc-based systems, although it isn't really too practical to use it with tape. The program will work with all printers connected via the Printer port (this is the normal practice), but will not work with printers that use the RS423 connector to avoid potential conflicts with other programs. It has also not proved possible to provide a spooler that will work concurrently with Wordwise, though there is no other problem in using it to print spooled Wordwise files.

The program occupies only one page of memory and, once loaded, can be used repeatedly. It makes use of the computer's "Event" system in order to interleave time allocated to spooling and time allocated to the foreground task in a way that makes both appear to take place concurrently. It is capable of running at the same time as other Event-driven routines, though this is partly dependent on how they are written.

SETTING UP THE SPOOLER

As with all programs that involve machine code, take great care when typing the program in, and make sure that you save it before running it in case the program should become corrupted. If you want to save some time when typing in the program, you can omit any text that follows a "\" character, as these are only comments.

Lines 1ØØ and 293Ø to 3ØØØ specify where in memory the program is to go, and the version supplied in the magazine contains a section of code to decide whether your Beeb is a disc or tape based machine, and to locate the code accordingly. If you find that the area allocated is not suitable, then by altering FNcode, in line 1ØØ, to a specific memory address (e.g. 1ØØ code=&9ØØ), then the program can be assembled to any area of your choice. Note that the tape/disc comment will no longer be valid, but the save addresses will be.

For disc, the program will assemble from address &AØØ onwards. As this is the cassette input buffer, this area is not normally used on a disc system, and so is a good place to hide this sort of code. However, you will sometimes find that this clashes with other programs which have the same idea. For instance, TOOLKIT uses Page &A.

You can avoid such clashes if you assemble the program in Page 9; this will, however, prevent your using speech, envelopes 5-16, the RS423 buffer (but the spooler doesn't support the serial port anyway) or the cassette output buffer. These may or not be serious limitations to you.

For tape, the code is assembled at &D00 onwards, which is only used by disc and other filing system ROMs. Again the alternative is Page &9 as described above.

When you set up the program, you must also set "linefeed", at line 1290, to fit your printer's requirements. Set it to FALSE if your printer provides its own linefeeds after a Carriage Return, or to TRUE if the computer has to supply it. If you normally need a *FX6,0 before you can print, your printer needs the extra linefeed.

Once you have entered and debugged the routine, you can save it directly as a machine-code file. To do this, run the program and then *SAVE it using the parameters as specified by the program itself (you can use an alternative to the name PRINT if you wish). When you load it in future, you only need to type either:

```
*LOAD PRINT
```

if you want to be able to CALL the spooler from within a Basic program, or:

```
*RUN PRINT
```

if you just want dump a single text file.

USING THE SPOOLER

The spooler always assumes that it is going to print an already-formatted (or pure ASCII) file; it cannot, for example, make sense of a Basic program which has been stored in its normal "tokenised" form. You must therefore prepare the file in advance by, for instance, *SPOOLing a program listing. If you are going to print a Wordwise file, use Option 8 to prepare it as spooled, formatted text, on disc or tape.

Once you have set up and saved the text you wish to print, load the spooler program, if it's not in memory already, and start it running by using a CALL command from Basic. For instance, if it has been assembled to Page &A, (as it would be for disc) use:

```
CALL &A20 <return>
```

If, on the other hand, you were using tape when you assembled it, use:

```
CALL &D20 <return>
```

If you have assembled the program into memory by running the original program (as listed), rather than loading it from tape or disc as a machine-code file, you have another way of calling the spooler. During assembly the computer is set up so that you can call the routine directly by the command:

```
*CODE <return>
```

The advantage is that you can run the program from within other languages. These usually allow you to give "*" commands but do not necessarily allow "CALL"s.

Whichever way you choose to call the spooler, the program will ask you for the name of the file you wish to print - enter it in the usual way and leave the spooler to do its job. You can carry on using your computer in almost any way you wish while the printer mutters away in the background. Every so often, the spooler will read another block of data from the file; while it is doing this, the foreground program will freeze, starting up again as soon as the next block of text has been loaded. The intermittent freezing is hardly noticeable if you are using discs, but is enough to cause inconvenience with tape.

It is for this reason that I only recommend the spooler for disc systems. Although it works perfectly well with tape, it cannot really be any more than a novelty, unless the foreground program is not bothered by frequent 1.5 second pauses. Don't try it with Planetoid! With a disc system, however, you can even read and write other data to and from disc while the spooler is busy.

POINTS TO NOTE
You should note that, unfortunately, the spooler is not compatible with Wordwise, so that although you can spool out Wordwise files without any problem at all, the spooler will not work from within Wordwise. This is a pity, since it would be useful to be able to continue to edit text etc while printing goes on, but is apparently unavoidable. It seems that Wordwise insists on closing the file that the spooler has opened, with the result that it cannot read the data!

When running, the routine ignores the Escape key, should it be pressed; this allows you to run programs which use this key for their own control. To stop the spooler in mid-print, you must press the Break key.

Despite its (relatively minor) limitations, the spooler is a very useful program. Once you have tried using it, I think that you will agree with me that it definitely falls into the "don't know how I managed before" category.

PROGRAM DESCRIPTION
The routine is driven by the 'vertical sync' event. The spooler should be compatible with the vast majority of programs, both Basic and machine-code. If you want to know more about events, I thoroughly recommend the article in BEEBUG Vol.2 No.6 (Nov 83). The spooler written in assembler, is contained in the procedure PROCassem. This first sets up all the constants and variables used by the spooler before determining the name of the file to be spooled.

Using the event mechanism, the spooler is called every 20 msec when it checks to see whether the next character can yet be sent to the printer. There will also be an occasional pause as the filing system loads the next data block from tape or disc, but this is unavoidable. This delay occurs when all of the data from the current block has been written to the printer. Once spooling is complete, the spool file is closed and the event mechanism disabled. The spooler will then remain dormant until called again. The program is well anotated throughout for those who are interested in its detailed workings.

```
   10 REM Program SPOOLER
   20 REM Version B1.1
   30 REM Author David Peckett
   40 REM Parallel printers only
   50 REM BEEBUG JULY 1984
   60 REM Program subject to Copyright
   70 :
  100 code=FNcode
  110 PROCassem
  120 ?USERV=init MOD 256
  130 REM SETUP *CODE ENTRY POINT
  140 USERV?1=init DIV 256
  150 PRINT'"Printer spooler installed."
  160 PRINT"Use *SAVE PRINT ";~code;" "
;~P%;" ";~(code+&20)
  170 PRINT"to save the spooler."
  180 PRINT"*CODE will spool when the B
asic program"
  190 PRINT"has just been RUN."
  200 PRINT"*RUN PRINT calls the spoole
r from ";
  210 IF code=&A00 PRINT"disc."' ELSE P
RINT"tape."'
  220 END
  230 :
 1000 DEF PROCassem
 1010 REM Set up constants
 1020 OSASCI=&FFE3
 1030 OSBGET=&FFD7
 1040 OSBYTE=&FFF4
 1050 OSFIND=&FFCE
 1060 OSWORD=&FFF1
 1070 EVNTV=&220
 1080 USERV=&200
 1090 REM Printer VIA addresses
 1100 ora=&FE61
 1110 pcr=&FE6C
 1120 ifr=&FE6D
 1130 ier=&FE6E
 1140 name=code:REM Space for file name
 1150 bufferparams=code+12:REM Buffer f
or name input
 1160 REM Program variables
 1170 temp=code+17
 1180 channel=code+18
 1190 tempier=code+19
 1200 temppcr=code+20
 1210 tempevntv=code+21
 1220 needlf=code+23
 1230 REM Set up info for name input
 1240 bufferparams!0=name
 1250 bufferparams?2=12:REM Max of 12 c
hars
 1260 bufferparams?3=32
 1270 bufferparams?4=126
 1280 REM Set flags
 1290 linefeed=FALSE:REM TRUE if printe
r needs linefeed
 1300 ?needlf=0
 1310 FOR pass=0 TO 2 STEP 2
 1320 P%=code+&20
 1330 [OPT pass
 1340 \
 1350 .init
 1360 LDX #LEN($message)
 1370 LDY #0
 1380 \
 1390 .type \ Print message
 1400 LDA message,Y
 1410 JSR OSASCI
 1420 INY
 1430 DEX
 1440 BNE type
 1450 \
 1460 \ Read file name
 1470 LDA #0
 1480 LDX #bufferparams MOD 256
 1490 LDY #bufferparams DIV 256
 1500 JSR OSWORD
 1510 BCC openfile \ Test for ESCAPE
 1520 RTS \ Exit if ESCAPE
 1530 \
 1540 .openfile
 1550 LDA #&40
 1560 LDX #name MOD 256
 1570 LDY #name DIV 256 \ Point to name
 1580 JSR OSFIND \ Open file
 1590 STA channel
 1600 BNE spoolstart \ Did it open?
 1610 RTS \ Return if not
 1620 \
 1630 \ Set up the Event mechanism
 1640 .spoolstart
 1650 SEI \ Block interrupts
 1660 LDA ier
 1670 STA tempier \ Save VIA mode..
 1680 LDA pcr
 1690 STA temppcr \..and handshake
 1700 LDA #&7F
 1710 STA ier \ Disable VIA interrupts
 1720 LDA #&0A
 1730 STA pcr \ Set handshake mode
 1740 LDA ora \ Dummy to start system
 1750 \ Save Event vector
 1760 LDA EVNTV:STA tempevntv
 1770 LDA EVNTV+1:STA tempevntv+1
 1780 LDA #spool MOD 256
 1790 STA EVNTV
 1800 LDA #spool DIV 256
 1810 STA EVNTV+1 \ Set spooler vector
 1820 LDA #14:LDX #4
 1830 JSR OSBYTE \ Enable vertical sync
Event
 1840 CLI \ Interrupts back on
 1850 RTS \ Back to user
 1860 \
 1870 \ Actual spooler routine
 1880 \
 1890 .spool
 1900 \ Save system variables
 1910 \ (Not absolutely essential:
 1920 \ see Beebug Vol.2 No.8,
 1930 \ but ensures routine
 1940 \ is re-entrant)
 1950 PHP:STA temp:PHA
 1960 TXA:PHA
 1970 TYA:PHA
 1980 LDA temp \ Event number
 1990 CMP #4 \ Was it vert. sync?
 2000 BNE exit \ if not
 2010 \ Monitor Printer VIA
 2020 LDA ifr \ Check printer flag
 2030 AND #2 \ Handshake complete?
 2040 BEQ exit \ If not, done
 2050 \
 2060 \ Disable Event 4 for file read
 2070 LDA #13:LDX #4
 2080 JSR OSBYTE
 2090 ]
 2100 :
 2110 REM Option for extra linefeed
 2120 REM to be sent to the printer.
 2130 REM Select appropriate bit
 2140 REM of code.
 2150
 2160 IF linefeed THEN PROCaddlf ELSE P
ROCnolf
 2170 :
 2180 REM Resume normal assembly
 2190 :
 2200 [OPT pass
 2210 \
 2220 .nolinefeed
 2230 \ Re-enable Event 4
 2240 LDA #14:LDX #4
 2250 JSR OSBYTE
 2260 JMP exit
 2270 \
 2280 .finished
 2290 \ Event 4 always disabled
 2300 \ when we get here
 2310 LDA #0:LDY channel
 2320 JSR OSFIND \Close file
 2330 SEI \ Block interrupts
 2340 LDA tempier
 2350 STA ier \ Restore VIA..
 2360 LDA temppcr
 2370 STA pcr \..as original
 2380 \ Restore Event vector
 2390 LDA tempevntv:STA EVNTV
 2400 LDA tempevntv+1:STA EVNTV+1
```

»

```
 2410 CLI \ Re-enable interrupts
 2420 \
 2430 .exit
 2440 PLA:TAY
 2450 PLA:TAX
 2460 PLA:PLP \ Restore system
 2470 \ Continue Event handling
 2480 JMP (tempevntv)
 2490 \
 2500 \ Message for file opening
 2510 \
 2520 .message
 2530 ]
 2540 $P%=CHR$7+"FILE? "
 2550 NEXT
 2560 ENDPROC
 2570 :
 2580 REM Code for when extra linefeed
needed
 2590 DEF PROCaddlf
 2600 [OPT pass
 2610 LDA needlf \ linefeed required th
is time?
 2620 BEQ getch \ If not, continue
 2630 LDX #0:STX needlf \ Clear flag
 2640 BEQ nogetch
 2650 \
 2660 .getch
 2670 \ Read a char from file
 2680 LDY channel
 2690 JSR OSBGET \ Get next char
 2700 BCS finished \ EOF?
 2710 \
 2720 .nogetch
 2730 STA ora \ Output to printer
 2740 CMP #13 \ Was it a CR?
 2750 BNE nolinefeed \ If not, continue
 2760 LDA #10:STA needlf \ Set flag
 2770 ]
 2780 ENDPROC
 2790 :
 2800 REM Code for printers with autoli
nefeed
 2810 DEF PROCnolf
 2820 [OPT pass
 2830 \ Get next char
 2840 LDY channel
 2850 JSR OSBGET
 2860 BCS finished \ EOF?
 2870 \
 2880 \ Print it
 2890 STA ora
 2900 ]
 2910 ENDPROC
 2920 :
 2930 DEF FNcode
 2940 LOCAL A%,Y%
 2950 A%=0:Y%=0
 2960 A%=(USR&FFDA)AND&FF0000 DIV &10000
 2970 IF A%=1 OR A%=2 Y%=&D00
 2980 IF A%=4 Y%=&A00
 2990 IF Y%=0 PRINT'"The spooler is des
igned to work with"'"tape or disc filin
g systems only!"':END
 3000 =Y%
```

Continued from page 17

```
 2460 B%=TIME:REPEATA%=INSTR("NBLJE",CH
R$(INKEY(0) AND&DF)):UNTILA% ORTIME>B%+
T%*100
 2470 IFA%=2 ANDPTR#A<2*L% THENUNTILFAL
SE ELSEIFA%=2 THENPTR#A=PTR#A-2*L%
 2480 IFA%=3 ORA%=4 THENPROCOpenCommand
:PRINT'"No of slides +/-?";:A$=E$:PROCI
nput(2):IFC=0 THENUNTILFALSE
 2490 C%=PTR#A+L%*(C-1):IFA%=3 ANDC%<=V%
 THENPTR#A=C% ELSEIFA%=3 THENUNTILFALSE
 2500 C%=PTR#A-L%*(C+1):IFA%=4 ANDC%>4
THENPTR#A=C% ELSEIFA%=4 THENUNTILFALSE
 2510 IFA%=5 ANDPTR#A=V%+L% THENUNTILFA
LSE ELSEIFA%=5 THENPTR#A=V%
 2520 UNTILTRUE
 2530 ENDPROC
 2540 :
 2550 DEF PROCNoOfColours
 2560 M%=3:IFINSTR("0346",STR$m%)>0 THE
NM%=1: REM Set the number of colours, M
%+1, according to the mode, m%.
 2570 ENDPROC
 2580 :
 2590 DEF PROCHeading
 2600 LOCALI%:FORI%=0TO1:PRINTTAB(10,I%
+1)CHR$141CR$CW$CY$" Slide Show   ";CZ$
:NEXT
 2610 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

LISTO ABBREVIATION

The BBC User Guide has a list (p.483) of keywords and their abbreviated forms, except for LISTO. It documents LIST as being shortened to L., so the conclusion that shortening LISTO to L.O would do the trick, is quite correct. For example L.O1 will list a program with a space inserted between the line number and the program text. (See User Guide, p.290 and remember to use a capital 'O' not zero here).

FREEZING AND SAVING SCREEN DISPLAYS

by Alan Webster

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

Have you ever wanted to save your Beeb's display in the middle of any program on to cassette or disc? Well now you have the chance with this event driven program, that combines a screen freezer and a routine to save the display.

As you may gather, there are two parts to this program. The first part is a screen freezer (based on the program by David Graham in BEEBUG Vol.2 No.7 Page 17), and the second part will optionally save the screen display. Both parts will work with most machine code programs, as well as with programs in Basic - so you can, for example, take screen dumps of machine code games to record high scores.

The program works by using the 'Key Pressed' event. When you press the @ key, a machine code routine at &D00 is executed. This freezes the Beeb until one of two things occurs.
1. Press the @ key again to return the Beeb to its normal state, or
2. Save the screen by pressing the minus (-) key.

After saving the screen, disc users may find that the computer 'locks up'. This is because some programs need the disc workspace to run, and will become corrupted.

Type the program in as listed and, before running, save a copy to cassette or disc. Then type:

```
RUN <return>
```

This sets up the machine code. Now you can load and run the program whose screen displays you wish to save. Please remember not to press Break at any time, otherwise you will have to start all over again.

When you have saved a screen on to cassette or disc, you can now reload that into memory by typing:

```
MODE n <return>
```

where n is the mode of the screen display, and then:

```
*LOAD SCREENx <return>
```

where x is a letter from A to Z, according to the filename that was saved.

Some of the programs that we could save displays from were:

Snooker	from Acornsoft
Swoop	from Program Power
Hunchback	from Superior Software
Slalom	from R H Software

PROGRAM NOTES

The program as listed is for cassette users. To alter the program to run on a disc system delete lines 345 and 435, and change line 100 to:

```
100 Code%=&A00
```

You can also change the address in line 100 to allow the freeze/save routine to reside in a different area of memory, if that specified area happens to be used by the game program.

The line at 590 is an Operating System command which *SAVEs the screen display as the file SCREENA. The first value after this is the start of screen memory and has to be altered appropriately for different screen modes.

Mode 7	&7C00
Mode 6	&6000
Modes 4 and 5	&5800
Mode 3	&4000
Modes 0, 1 and 2	&3000

Once the file SCREENA has been saved, the program automatically updates the filename to SCREENB, and then to SCREENC and so on.

One final note about saving on cassette. There will not be any visible signs that the Beeb is saving the screen as all messages are temporarily disabled. You should press 'Play' and 'Record' on your cassette recorder immediately before pressing '-' to save the screen, though you can, however, watch the cassette motor LED to see when it has finished recording.

```
 1Ø REM PROGRAM Freezer / Saver
 2Ø REM AUTHOR  Alan R Webster
 3Ø REM VERSION B1.1
 4Ø REM BEEBUG  JULY 1984
 5Ø REM PROGRAM SUBJECT TO COPYRIGHT
 6Ø:
1ØØ Code%=&DØØ
11Ø osc=Code%+&8Ø
12Ø file=osc+11
13Ø FOR T%=Ø TO 3 STEP 3
14Ø P%=Code%
15Ø [OPT T%
16Ø .begin
17Ø PHP:PHA
18Ø TXA:PHA
19Ø TYA:PHA
2ØØ CPY #ASC "@"
21Ø BNE exit
22Ø .rell
23Ø JSR key
24Ø BNE rell
25Ø LDA #129
26Ø LDX #232
27Ø LDY #255
28Ø JSR &FFF4
29Ø CPY #255
3ØØ BEQ save
31Ø JSR key
32Ø BNE exit
33Ø JMP rell
34Ø .save
35Ø LDA #139
36Ø LDX #1
37Ø LDY #Ø
38Ø JSR &FFF4
39Ø CLC
4ØØ LDX #osc AND 255
41Ø LDY #osc DIV 256
42Ø LDA #Ø
43Ø JSR &FFF7
44Ø CLC
45Ø LDA file
46Ø ADC #1
47Ø STA file
48Ø LDA #139
49Ø LDX #1
5ØØ LDY #1
51Ø JSR &FFF4
52Ø .exit
53Ø PLA:TAY
54Ø PLA:TAX
55Ø PLA:PLP
56Ø RTS
57Ø .key
58Ø LDA #129
59Ø LDX #184
6ØØ LDY #255
61Ø JSR &FFF4
62Ø TYA
63Ø RTS
64Ø ]
65Ø NEXT
66Ø:
67Ø C$="SAVE SCREENA 7CØØ 8ØØØ"
68Ø C$=C$+CHR$13
69Ø $osc=C$
7ØØ ?&22Ø=Code% AND 255
71Ø ?&221=Code% DIV 256
72Ø *FX14,2
73Ø END
```

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOA

TESTING OUT YOUR MICRO

This month's issue of BEEBUG carries an in-depth review of Acorn's 65Ø2 Second Procesor. As a result, space is very much at a premium, and the next part in our series 'Tesing Out Your Micro' has had to be carried over to next month.

HINT WINNERS

This month the £1Ø prize goes to A.Porter and the £5 prize to A.Benham. Thanks for all your hints. Keep them rolling in (what about Second Processors?).

MAGAZINE CASSETTE AND DISC

This month all the programs in this issue of BEEBUG are available on both cassette and disc. Each of the string handling routines from the BEEBUG Workshop has been expanded to provide a full demonstration program in each case. The magazine disc will cost £4.75 and there are arrangemnts for subscribers to the magazine cassette to convert this to a disc subscription. See the member's price list for full details.

ACORN Z8Ø SECOND PROCESSOR

Hard on the heels of their 65Ø2 Second Processor, Acorn have now launched the Z8Ø Second Processor. This is a comprehensive small business system, with many applications packages included in the price of £299 (plus VAT), including word processing, database and spreadsheet. The system uses the well known CP/M operating system to provide access to a wealth of business software. We hope to include a review of this next month.

ACORNSOFT'S VIEW AND VIEWSHEET

The Viewsheet package, reviewed in this issue, should be availble in ROM format from early July. At the same time Acornsoft will be releasing Version 2.1 of the View word processing package to supercede the previous version 1.4. More information on View next month.

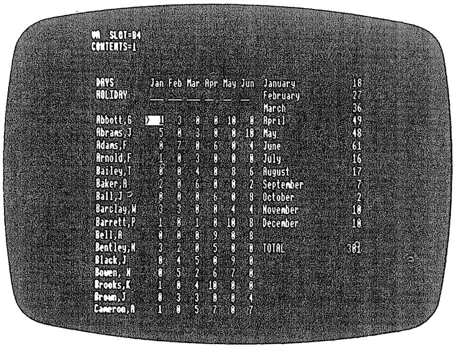
VIEWSHEET AND ULTRACALC TWO SPREADSHEET PACKAGES FOR THE BEEB

Reviewed by David Otley

Following the article last month, which described the function of spreadsheet programs, David Otley continues with a review of two packages that are likely to be major contenders in the spreadsheet market for the Beeb.

Ultracalc
Supplier: BBC Soft
Price : £74.75 inc VAT

ViewSheet
Supplier: Acornsoft
Price : £59.8Ø inc VAT



```
   SLOT=B4
CONTENTS=1

DAYS        Jan Feb Mar Apr May Jun  January     18
HOLIDAY     ___ ___ ___ ___ ___ ___  February    27
                                     March       36
Abbott,G  >   1   3   0   0  10   0  April       49
Abrams,J      5   0   3   0   0  10  May         48
Adams,F       0   7   0   6   0   4  June        61
Arnold,F      1   0   3   0   0   0  July        16
Bailey,T      0   0   4   0   8   6  August      17
Baker,A       2   0   6   0   0   2  September    7
Ball,J        0   0   0   6   0   0  October      2
Barclay,W     3   3   0   0   4   4  November    10
Barrett,P     1   0   1   0  10   8  December    10
Bell,A        0   0   0   9   0   8
Bentley,N     3   2   0   5   0   0  TOTAL      301
Black,J       0   4   5   0   9   0
Bowen, W      0   5   2   6   7   0
Brooks,K      1   0   4  10   0   0
Brown,J       0   3   3   0   0   4
Cameron,A     1   0   5   7   0   7
```

INTRODUCTION

Both ViewSheet and Ultracalc are supplied on 16K EPROMs and in this form, are only suitable for use on Model B machines. They both come supplied in similar packages; A5 boxes, each with a stout manual and the ROM containing the spreadsheet program itself. As the version of ViewSheet, kindly loaned to BEEBUG by Acornsoft, was a pre-release copy, it is not possible to comment on the quality of the documentation due to its incomplete nature. Previous releases, like the View word processor package, have had accompanying manuals and text that have been excellent. Hopefully, one can expect the same in this instance.

The manual for Ultracalc was very good indeed. However, the packaging was a little suspect for such a costly item of software. When the copy for review arrived, the ROM was loose inside the box and required delicate surgery to be performed on its legs before being used. Selection is made with a '*CALC' command. (Similarly for ViewSheet but the command is '*SHEET').

A COMPARISON

Last month's article introduced the important principles of spreadsheet packages and you may wish to refresh your memory on this before continuing to read this review.

ViewSheet offers a potentially larger worksheet (255 rows x 255 columns) than Ultracalc (255 rows by 63 columns), but as the available memory limits total model size to around 16ØØ cells (Mode 7), this is hardly significant.

Much more important is the capability of ViewSheet to operate in any screen mode, whereas Ultracalc is fixed in the 4Ø character width of Mode 7. This limits the number of columns that can be displayed on the screen to five or six. In Mode 3, ViewSheet can cope with up to twelve columns satisfactorily, although a monitor is essential for clarity. The ability to see a good-sized chunk of a sheet is useful, especially when it is large; unfortunately, in Mode 3 ViewSheet has limited memory available (I just managed to squeeze a 17 column by 36 row model into Mode 3), making a second processor a desirable addition [ViewSheet does work across the tube - see the 65Ø2 2nd processor review elsewhere in this issue-Ed].

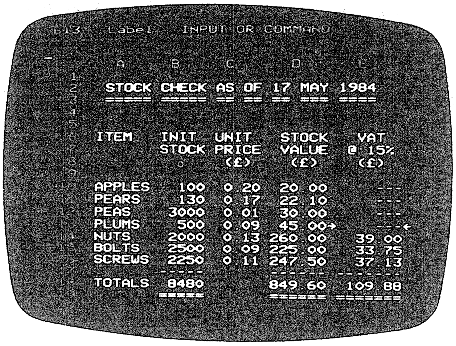
However, Ultracalc allows column widths to be specified individually, whereas ViewSheet requires all columns to be the same width. This restriction is particularly irksome, although it can be partially compensated for by the ability to set row labels into the screen border (itself of variable width), and by the use of screen windows.

These 'windows' are unique to ViewSheet and allow up to ten different sections of the spreadsheet to be displayed on the screen at once, each in their own format. Thus, on a large sheet, only data entry and final result rows might be displayed, allowing the impact of changes in data to be observed without having to move the screen window. Windows are also used in order to define areas of the model to be printed out. Ultracalc is less complicated in this respect allowing only one specified rectangular block of the sheet to be printed at a time, and so is easier to use with simple applications.

Ultracalc also permits individual entries to be made in different foreground and background colours - useful in highlighting important results - and negative numbers may be optionally displayed in red. Both programs allow negative numbers to be shown in brackets rather than with a minus sign, again indicative of the business applications envisaged.

The screen borders and data entry line are displayed in green by Ultracalc. This is acceptable on a colour display but slightly dim on a monochrome screen. It would appear that Ultracalc is better suited to the user with a colour TV while ViewSheet is comfortably used with any kind of monitor, monochrome included. Its display is white on black, but both foreground and background colours can be altered in all modes but Mode 7. This is particularly useful in Mode 3 where changing the main background colour leaves lines across the screen between rows which help to guide the eye.

For some reason the two programs calculate in different directions. ViewSheet calculates row by row, whereas Ultracalc goes column by column. This usually doesn't matter, but both methods have their advantages in specific applications (eg. when totals or closing balances from one row/col have to be transferred to form



```
E13   Label   INPUT OR COMMAND
-
        A      B      C       D       E
 1
 2  STOCK CHECK AS OF 17 MAY 1984
 3  ===== ===== == == == === ====
 4
 5
 6  ITEM    INIT  UNIT    STOCK    VAT
 7          STOCK PRICE   VALUE   @ 15%
 8            o    (£)     (£)     (£)
 9
10  APPLES   100  0.20   20.00      ---
11  PEARS    130  0.17   22.10      ---
12  PEAS    3000  0.01   30.00      ---
13  PLUMS    500  0.09   45.00→     ---←
14  NUTS    2000  0.13  260.00    39.00
15  BOLTS   2500  0.09  225.00    33.75
16  SCREWS  2250  0.11  247.50    37.13
            -----       ------  -------
18  TOTALS  8480        849.60   109.88
            =====       ======  =======
```

the opening figure in the next row/col). Most spreadsheets give the user the option of specifying which method should be used; not to have this capability is annoying, although alternative sheet layouts can be devised to cope with this problem.

Despite their many similar facilities, the programs feel rather different to use. ViewSheet makes extensive use of the function keys in a similar way to the View word processor. In Ultracalc commands are entered more directly, prefaced by a '/' character (familiar to users of Supercalc) to distinguish them from values or labels. In terms of setting up a spreadsheet to perform a specific job, Ultracalc has slightly more versatile facilities in replication, display formatting and model protection. It also has a 'net present value' (NPV) command invaluable to those involved in financial modelling but which inexplicably, is not mentioned in the manual, except in a summary list of commands. Finally Ultracalc's lookup tables allow for text as well as numeric entries.

Although both programs can deal with similarly sized models (around 1600 cells in Mode 7), Ultracalc is slower in recalculation (15secs compared with 5 secs) and much slower in saving to and loading from disc (greater than 60 secs compared with less than 10 secs for a maximum sized model on ViewSheet). However, an overwhelming advantage of ViewSheet is that any selected screen or printer output can be written to a data file and transferred into a word processor for incorporation into a report or other document. It is designed to be compatible with the View word processor, but files can also be transferred into Wordwise (although I would suggest that final formatting of the text is completed before inserting ViewSheet tables due to a slight problem with the ends of lines). As far as I could establish, there is no similar facility available in Ultracalc.

CONCLUSIONS

In many ways Ultracalc is the more attractive program, especially for the home user. The commands are logical and easy to use without reference to key headings. It has most of the operational features of ViewSheet and a few useful extensions (variable column width, block copying, protection and formatting, lookup tables using labels and an NPV function). But it has two severe disadvantages.

Firstly, it uses only a 4Ø character screen which drastically limits the amount of a sheet that can be viewed at one time. However, it should be noted that ViewSheet has the same limitation for models using more than 9K of memory (unless a second processor is used).

Secondly, it has no facility for transferring results directly into a word processor. Here ViewSheet, with its highly versatile screen and print windows, can produce almost any type of display that is required, albeit with some necessary set-up effort, and is clearly superior.

Both programs are adequate for serious use and represent an important addition to available software. They are both comparable with Visicalc, although it is perhaps surprising that they offer no substantial advance on its capabilities (probably due to memory limitations). Despite the internal advantages of Ultracalc, the display and file transfer advantages of ViewSheet would sway the balance for me. If Acornsoft now produce the promised data-base management system in a form that integrates with both View and ViewSheet, then BBC micro users will have a total data management package that compares well with that available for any micro in its class.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

GRAPHICS EXTENSION ROM VARIABLES - M. Chang Sing Pang

Use LOCAL variables within procedures and functions to drive the Computer Concepts Graphics ROM from Basic to ensure that system variables A% to E% are not interfered with by your program. This is necessary because the GROM makes use of them itself.

PHASE SELECTION IN MICROPOWER'S "CROAKER" - J.Brunyee & J.Spinks

If you find Program Power's Croaker too easy to start with, then during the part of the program that displays which phase of the game you are starting, press Escape as many times as the phase that you wish to play.

RELOCATE PROGRAM FUNCTION KEY - M.C.Behrend

This procedure can be added to a program to move itself down, when it's run, and back up again, afterwards. If this is stored, using *SPOOL, as a text file it can then be merged into a program with *EXEC. Note that it makes use of the variable Z% for the duration of the program. (As listed, it will relocate to and from &EØØ).

```
    1 PROCdown(&EØØ)
    32765 DEFPROCup(L%)P%=PA.:T%=TOP:D%=P%-L%:U%=T%-D%:IFD%<ØFORM%=U%-U%MOD4TOL%STEP
    -4:!M%=M%!D%:NEXT:PA.=L%:END
    32766 PROCup(Z%)
    32767 DEFPROCdown(L%)P%=PA.:T%=TOP:D%=P%-L%:U%=T%-D%:IFD%>ØFORM%=L%TOU%STEP4:!M%
    =M%!D%:NEXT:Z%=P%:PA.=L%:?18=U%MOD256:?19=U%DIV256:RUN:ELSEENDPROC
```

Your own program should occupy lines between 1 and 32765.

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

SNIP SNAP (32K)

by D.J. Pilling

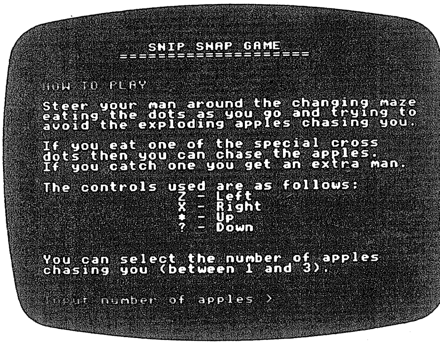
In recent months, we have published two splendid games from D.J.Pilling. He has now produced a very different game for your delight and entertainment. SNIP SNAP is great fun, and quite short to type in.

The game of SNIP SNAP has a number of features typical of many computer games of the 'Snapper' type. You have to guide a 'man' around the screen, eating up the dots as you go while trying to avoid the exploding apples chasing after you. What adds an extra dimension of thought to this game is the way the grid changes at random intervals between horizontal rows and vertical columns. One moment you can feel quite safe, and the next an apple (hardly Golden Delicious despite the colour) is poised to destroy you.

If you manage to eat one of the special 'cross' dots, you can then chase and destroy the apples (shown by a change of screen colour) but be ready to run away again as soon as the screen reverts to its normal colour. Destroy the apples and you gain extra lives.

The program should be typed in with care and saved to cassette or disc. When run, it displays full instructions on the screen. The main controls, as usual, are 'Z' and 'X' for left and right, and '*' and '?' for up and down. Have fun (and send us your high score too).

NOTE FOR DISC USERS
You will need to set PAGE to &1200 before loading and running SNIP SNAP, or use *MOVE 1200 from TOOLKIT, or make use of the 'movedown' routine printed elsewhere in this issue.

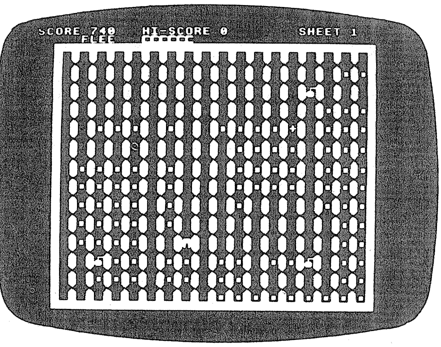


```
              SNIP SNAP GAME
            ==================

HOW TO PLAY

Steer your man around the changing maze
eating the dots as you go and trying to
avoid the exploding apples chasing you.

If you eat one of the special cross
dots then you can chase the apples.
If you catch one you get an extra man.

The controls used are as follows:
               Z - Left
               X - Right
               * - Up
               ? - Down


You can select the number of apples
chasing you (between 1 and 3).

Input number of apples >
```

```
   10 REM PROGRAM SNIP SNAP
   20 REM VERSION B0.4
   30 REM AUTHOR  D.J.PILLING
   40 REM BEEBUG  JULY 1984
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 ON ERROR GOTO 410
  110 MODE1
  120 DIM D% 1200
  130 DIM ML$(1),MR$(1),MU$(1),MD$(1),D
$(2)
  140 DIM A%(2),B%(2)
  150 PROCCHAR:PROCS:PROCI
  160 REPEAT
  170 REPEAT
  180 SH%=SH%+1:PROCT
  190 S%=1
  200 UP%=10-SH%:IF UP%<2 UP%=2
  210 IF N%=0 M1%=-1:M2%=-1:GOTO240
  220 IF N%=1 M1%=UP%DIV2:M2%=-1:GOTO240
  230 M1%=(UP%+1)DIV3:M2%=2*M1%
  240 REPEAT
  250 PROCCM(0)
  260 FOR L%=0 TO UP%
  270 IF FNH PROCE:L%=UP%
  280 PROCMAN
  290 IF L%=M1% PROCCM(1)
  300 IF L%=M2% PROCCM(2)
  310 K%=K%-1
  320 IF K%>0 GOTO340
  330 S%=-S%:K%=10+RND(5)*5:SOUND3,-15,
40,3:SOUND3,-15,80,4:IF S%>0 VDU19,1,0,
0,0,0:VDU19,2,1,0,0,0 ELSE VDU19,1,1,0,
0,0:VDU19,2,0,0,0,0
  340 NEXT
  350 UNTIL OVER%
  360 CLS
  370 UNTIL MEN%=0
  380 UNTIL FNEG
  390 MODE 7
  391 END
  400 :
  410 ON ERROR OFF
```

»

```
  42Ø MODE 7:IF ERR=17 END
  43Ø REPORT:PRINT" at line ";ERL
  44Ø END
  45Ø :
 1ØØØ DEFPROCCHAR
 1Ø1Ø VDU23,241,255,126,6Ø,8,16,6Ø,126,
255
 1Ø2Ø VDU23,242,Ø,255,255,255,255,255,2
55,Ø
 1Ø3Ø VDU23,243,255,255,255,255,255,255
,255,255
 1Ø4Ø VDU23,244,31,16,16,124,254,254,25
4,124
 1Ø5Ø VDU23,246,Ø,Ø,6Ø,6Ø,6Ø,6Ø,Ø,Ø
 1Ø6Ø VDU23,247,66,195,195,195,195,255,
255,126
 1Ø7Ø VDU23,248,126,255,255,195,195,195
,195,66
 1Ø8Ø VDU23,249,126,255,224,224,224,224
,255,126
 1Ø9Ø VDU23,25Ø,126,255,7,7,7,7,255,126
 11ØØ VDU23,251,1Ø2,231,231,231,255,255
,255,126
 111Ø VDU23,252,126,255,255,255,231,231
,231,1Ø2
 112Ø VDU23,253,126,255,255,24Ø,24Ø,255
,255,126
 113Ø VDU23,254,126,255,255,15,15,255,2
55,126
 114Ø ENDPROC
 115Ø :
 116Ø DEFPROCT
 117Ø CLS
 118Ø VDU23,1,Ø;Ø;Ø;Ø;
 119Ø VDU19,3,3,Ø,Ø,Ø:COLOUR3
 12ØØ VDU19,1,Ø,Ø,Ø,Ø:VDU19,2,1,Ø,Ø,Ø
 121Ø PRINTTAB(2,2)STRING$(37,CHR$243)
 122Ø PRINTTAB(2,3Ø)STRING$(37,CHR$243)
 123Ø FOR I%=3TO29:PRINTTAB(2,I%)CHR$24
3;:PRINTTAB(38,I%)CHR$243;:NEXT
 124Ø FORI%=3TO29 STEP2:PRINTTAB(3,I%)S
TRING$(35,CHR$246):NEXT
 125Ø COLOUR2:COLOUR129:FOR I%=4TO36STE
P2:PRINTTAB(I%,3)STRING$(13,CHR$243+CHR
$8+CHR$1Ø+CHR$241++CHR$8+CHR$1Ø)CHR$243
:NEXT:COLOUR1:COLOUR128
 126Ø FOR I%=3TO37STEP2:PRINTTAB(I%,4)S
TRING$(13,CHR$242+CHR$8+CHR$1Ø+CHR$1Ø):
NEXT
 127Ø COLOUR3
 128Ø VDU19,1,Ø,Ø,Ø,Ø
 129Ø M%=1:X%=3:Y%=3:PRINTTAB(X%,Y%)ML$
(1);
 13ØØ RESTORE
 131Ø FOR Z%=ØTON%:READ A%(Z%),B%(Z%):P
RINTTAB(A%(Z%),B%(Z%))CHR$244;:NEXT
 132Ø FOR I%=ØTO1147STEP4:!(D%+I%)=&Ø1Ø
1Ø1Ø1:NEXT
```



```
 133Ø FOR K%=ØTO4:I%=3+2*RND(16):J%=5+2
*RND(12):?(D%+I%+J%*37)=2:PRINTTAB(I%,J
%)"+":NEXT
 134Ø SS%=SC%
 135Ø PRINTTAB(Ø,Ø)"SCORE ";SC%;TAB(12)
"HI-SCORE ";HSC%;TAB(3Ø)"SHEET ";SH%;
 136Ø PRINTTAB(5,1)"FLEE "
 137Ø IF MEN%>28 I%=28 ELSE I%=MEN%
 138Ø PRINTTAB(12,1)STRING$(I%,CHR$249);
 139Ø OVER%=FALSE:CF%=FALSE
 14ØØ ENDPROC
 141Ø :
 142Ø DATA37,3,37,29,3,29
 143Ø :
 144Ø DEFPROCS
 145Ø ML$(Ø)=CHR$25Ø:ML$(1)=CHR$254
 146Ø MR$(Ø)=CHR$249:MR$(1)=CHR$253
 147Ø MU$(Ø)=CHR$247:MU$(1)=CHR$251
 148Ø MD$(Ø)=CHR$248:MD$(1)=CHR$252
 149Ø D$(Ø)=" ":D$(1)=CHR$246:D$(2)="+"
 15ØØ HSC%=Ø:SC%=Ø:MEN%=3:SH%=Ø:SS%=Ø
 151Ø ENDPROC
 152Ø :
 153Ø DEFPROCMAN
 154Ø FORI%=1TO2ØØ:NEXT
 155Ø IF M%=Ø M%=1 ELSE M%=Ø
 156Ø IF S%=1 GOTO 16ØØ
 157Ø IFINKEY-67 IF X%<36 PRINTTAB(X%,Y
%)" ":X%=X%+2:PRINTTAB(X%,Y%)MR$(M%):SO
UND1,-15,53,1
 158Ø IFINKEY-98 IF X%>4 PRINTTAB(X%,Y%
)" ":X%=X%-2:PRINTTAB(X%,Y%)ML$(M%):SOU
ND1,-15,53,1
 159Ø GOTO162Ø
 16ØØ IFINKEY-73 IF Y%>4 PRINTTAB(X%,Y%
)" ":Y%=Y%-2:PRINTTAB(X%,Y%)MU$(M%):SOU
ND1,-15,53,1
 161Ø IFINKEY-1Ø5 IF Y%<29 PRINTTAB(X%,
Y%)" ":Y%=Y%+2:PRINTTAB(X%,Y%)MD$(M%):S
OUND1,-15,53,1
```

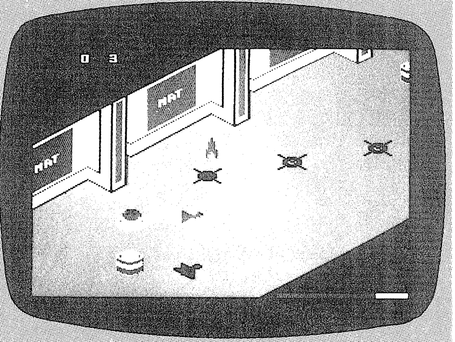
»

```
 162Ø IF ?(D%+X%+Y%*37)=1 SC%=SC%+5:?(D
%+X%+Y%*37)=Ø:PRINTTAB(Ø,Ø)"SCORE ";SC%
; ELSE IF ?(D%+X%+Y%*37)=2 ?(D%+X%+Y%*3
7)=Ø:IF NOT CF% PROCCHASE
 163Ø IFSC%-SS%=1235 IFNOTOVER%:OVER%=T
RUE:SOUND3,-15,2Ø,1Ø
 164Ø ENDPROC
 165Ø :
 166Ø DEFPROCCM(Z%)
 167Ø IF S%=1 GOTO171Ø
 168Ø IF A%(Z%)<X% PRINTTAB(A%(Z%),B%(Z
%))D$(?(D%+A%(Z%)+B%(Z%)*37)):A%(Z%)=A%
(Z%)+2:PRINTTAB(A%(Z%),B%(Z%))CHR$244:S
OUND2,-15,2Ø,1:ENDPROC
 169Ø IF A%(Z%)>X% PRINTTAB(A%(Z%),B%(Z
%))D$(?(D%+A%(Z%)+B%(Z%)*37)):A%(Z%)=A%
(Z%)-2:PRINTTAB(A%(Z%),B%(Z%))CHR$244:S
OUND2,-15,2Ø,1:ENDPROC
 17ØØ ENDPROC
 171Ø IF B%(Z%)<Y% PRINTTAB(A%(Z%),B%(Z
%))D$(?(D%+A%(Z%)+B%(Z%)*37)):B%(Z%)=B%
(Z%)+2:PRINTTAB(A%(Z%),B%(Z%))CHR$244:S
OUND2,-15,2Ø,1:ENDPROC
 172Ø IF B%(Z%)>Y% PRINTTAB(A%(Z%),B%(Z
%))D$(?(D%+A%(Z%)+B%(Z%)*37)):B%(Z%)=B%
(Z%)-2:PRINTTAB(A%(Z%),B%(Z%))CHR$244:S
OUND2,-15,2Ø,1:ENDPROC
 173Ø ENDPROC
 174Ø :
 175Ø DEFFNH=(A%(Ø)=X%ANDB%(Ø)=Y%)OR(A%
(1)=X%ANDB%(1)=Y%)OR(A%(2)=X%ANDB%(2)=Y
%)
 176Ø :
 177Ø DEFPROCE
 178Ø SOUNDØ,-15,4,25
 179Ø PRINTTAB(5,1)"BANG":MEN%=MEN%-1:O
VER%=TRUE:SS%=SC%
 18ØØ GCOLØ,3
 181Ø FOR V%=ØTO2Ø:MOVE X%*32+16,1ØØ8-Y
%*32:DRAW (X%+4-RND(6))*32+16,1ØØ8-(Y%+
4-RND(6))*32:NEXT
 182Ø TX%=TIME:REPEAT UNTIL TIME>TX%+1ØØ
 183Ø ENDPROC
 184Ø :
 185Ø DEFPROCCHASE
 186Ø SOUND3,-15,4Ø,3:SOUND3,-15,6Ø,2:S
OUND3,-15,8Ø,2
 187Ø CF%=TRUE:PRINTTAB(5,1)"CHASE"
 188Ø K%=Ø:L%=UP%:KA%=Ø
 189Ø FOR G%=ØTO15
 19ØØ PROCFM(Ø)
 191Ø FOR LA%=Ø TO UP%:IF KA%>Ø GOTO194Ø
 192Ø KA%=1Ø+RND(5)*5:S%=-S%:IF S%>Ø VD
U19,1,Ø,Ø,Ø,Ø:VDU19,2,4,Ø,Ø,Ø ELSE VDU1
9,1,4,Ø,Ø,Ø:VDU19,2,Ø,Ø,Ø,Ø
 193Ø SOUND3,-15,2Ø,4:SOUND3,-15,7Ø,3
 194Ø IF LA%=M1% PROCFM(1)
 195Ø IF LA%=M2% PROCFM(2)
 196Ø PROCMAN:KA%=KA%-1
 197Ø IF FNH PROCW:LA%=UP%:G%=15
 198Ø NEXT:NEXT
 199Ø CF%=FALSE:PRINTTAB(5,1)"FLEE  "
 2ØØØ ENDPROC
 2Ø1Ø :
 2Ø2Ø DEFPROCFM(Z%)
 2Ø3Ø IF S%=1 GOTO2Ø7Ø
 2Ø4Ø IF A%(Z%)<=X% IF A%(Z%)>4 PRINTTA
B(A%(Z%),B%(Z%))D$(?(D%+A%(Z%)+37*B%(Z%
))):A%(Z%)=A%(Z%)-2:PRINTTAB(A%(Z%),B%(
Z%))CHR$244:SOUND2,-15,1ØØ,1:ENDPROC
 2Ø5Ø IF A%(Z%)>=X% IF A%(Z%)<36PRINTTA
B(A%(Z%),B%(Z%))D$(?(D%+A%(Z%)+37*B%(Z%
))):A%(Z%)=A%(Z%)+2:PRINTTAB(A%(Z%),B%(
Z%))CHR$244:SOUND2,-15,1ØØ,1:ENDPROC
 2Ø6Ø ENDPROC
 2Ø7Ø IF B%(Z%)<=Y% IF B%(Z%)>4PRINTTAB
(A%(Z%),B%(Z%))D$(?(D%+A%(Z%)+37*B%(Z%)
)):B%(Z%)=B%(Z%)-2:PRINTTAB(A%(Z%),B%(Z
%))CHR$244:SOUND2,-15,1ØØ,1:ENDPROC
 2Ø8Ø IF B%(Z%)>=Y% IF B%(Z%)<29PRINTTA
B(A%(Z%),B%(Z%))D$(?(D%+A%(Z%)+37*B%(Z%
))):B%(Z%)=B%(Z%)+2:PRINTTAB(A%(Z%),B%(
Z%))CHR$244:SOUND2,-15,1ØØ,1:ENDPROC
 2Ø9Ø ENDPROC
 21ØØ :
 211Ø DEFPROCW
 212Ø SOUNDØ,-15,4,2Ø:SOUND1,Ø,Ø,2Ø:SOU
ND1,-15,53,1Ø:SOUND1,-15,65,1Ø:SOUND1,-
15,85,1Ø
 213Ø MEN%=MEN%+1:PRINTTAB(5,1)"CAUGHT"
 214Ø IF MEN%<29 PRINTTAB(11+MEN%,1)CHR
$249;
 215Ø TX%=TIME:REPEAT  UNTIL TIME>TX%+3
ØØ
 216Ø RESTORE
 217Ø FOR Z%=ØTON%:PRINTTAB(A%(Z%),B%(Z
%))" ":READ A%(Z%),B%(Z%):PRINTTAB(A%(Z
%),B%(Z%))CHR$244;:NEXT
 218Ø X%=3:Y%=3:PRINTTAB(3,3)MR$(M%)
 219Ø ENDPROC
 22ØØ :
 221Ø DEFFNEG
 222Ø CLS:PRINTTAB(5,5)"GAME OVER"
 223Ø PRINT'TAB(5)"YOUR SCORE -> ";SC%
 224Ø PRINT'TAB(5)"HIGH SCORE -> ";HSC%
 225Ø IF SC%>HSC% HSC%=SC%
 226Ø SC%=Ø:MEN%=3:SH%=Ø:SS%=Ø
 227Ø *FX15,Ø
 228Ø PRINT'''TAB(5)"ANOTHER GAME Y/N ";
 229Ø I$=GET$
 23ØØ IF I$="Y" =FALSE ELSE IF I$="N" =
TRUE ELSE GOTO229Ø
 231Ø DEFPROCI
 232Ø VDU19,3,6,Ø,Ø,Ø
 233Ø VDU19,1,2,Ø,Ø,Ø
 234Ø CLS:COLOUR3:PRINT''TAB(11)"SNIP S
NAP GAME"
 235Ø PRINTTAB(8)"=================="
''
 236Ø COLOUR2
 2370 PRINT"HOW TO PLAY"':COLOUR 1
 2380 PRINT"Steer your man around the c
hanging maze"'"eating the dots as you g
o and trying to"'"avoid the exploding a
pples chasing you."
 2390 PRINT'"If you eat one of the spec
ial cross"'"dots then you can chase the
 apples."'"If you catch one you get an
extra man."'
 2400 PRINT"The controls used are as fo
llows:"'TAB(14)"Z - Left"'TAB(14)"X - R
ight"'TAB(14)"* - Up"'TAB(14)"? - Down"
''
 2410 PRINT"You can select the number o
f apples"'"chasing you (between 1 and 3
)."'
 2420 COLOUR2
 2430 PRINT'"Input number of apples > ";
 2440 N%=GET-ASC"0"
 2450 IF N%>0 AND N%<4 N%=N%-1 ELSE GOT
O2440
 2460 ENDPROC
```

FORTRESS - A SUPERB GAME FROM PACE

Reviewed by Hugh Brown-Smith

Title	: Fortress
Supplier	: Pace
Price	: £9.95
Rating	: ****



Fortress is advertised as being a 3D game and is one of the few games on the market that lives up to this description. Its arcade equivalent is Zaxxon and the faithfulness of the computer to the original is exceptional throughout the initial screens. The only features missing are the swarms of killer planes.

For those not familiar with Zaxxon, the game presents a plane which you have to fly across a realistically three dimensional landscape. The screen scrolls diagonally giving the walls, fuel dumps and other obstacles a striking depth. The game is made more difficult by the absence of any altitude indicator, so that you must rely on the position of your shadow to judge continually your own height above the ground and other objects.

Rockets are fired up at you as you pass over, and missile bases will also attack, making survival a difficult task, while there are still homing missiles and force fields to come. Your precious fuel will disappear at an alarming rate, so that to finish even the first layer you will need to shoot nearly everything in sight. Additional layers or screens present new hazards to tax your hard won skills.

All in all, this is an excellent game in full colour and with realistic sound, good enough to keep even the keenest fighter pilot happy for hours. Only the lack of killer planes, as in the original arcade game, prevent me from awarding Fortress five stars.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CONTEMPORARY IMPROVISATIONS - R.Tobin

Type G.0:G.0:G.0:G.0:G.0:....etc., until you've filled the buffer (about 6 lines in Mode 7). Then hit return. You will of course get the error 'No such line' but it also has other effects...

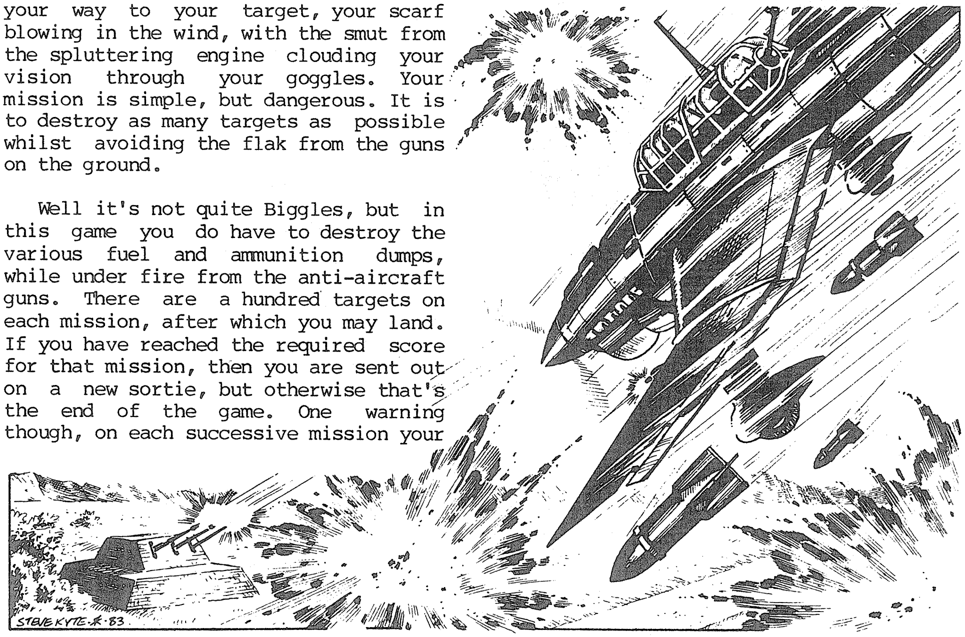
Tested on Basic I & II and O.S. 1.2

DIVE BOMBER (32K)

by N. Mallinson

Just imagine the thrill as you speed your way to your target, your scarf blowing in the wind, with the smut from the spluttering engine clouding your vision through your goggles. Your mission is simple, but dangerous. It is to destroy as many targets as possible whilst avoiding the flak from the guns on the ground.

Well it's not quite Biggles, but in this game you do have to destroy the various fuel and ammunition dumps, while under fire from the anti-aircraft guns. There are a hundred targets on each mission, after which you may land. If you have reached the required score for that mission, then you are sent out on a new sortie, but otherwise that's the end of the game. One warning though, on each successive mission your ceiling will not be as high as it was before, forcing a lower and more hazardous approach to each target.



You have three controls:

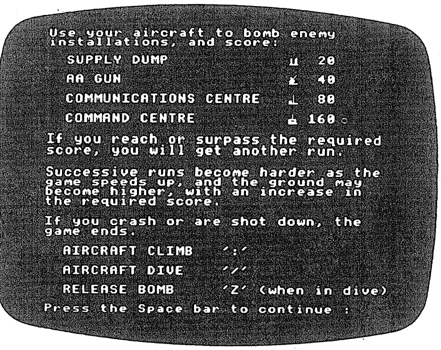
: - up
/ - down
z - release bomb.

Note that bombs may only be released once the plane has achieved a good dive. You will find that bombing runs can be made more accurately from a low level approach, but then you will have to be nimble fingered to avoid crashing; and the anti-aircraft batteries are a good deal more accurate when you are flying low.

Good luck Biggles!

Take care when typing the program into your micro, particularly with the section of assembler code included from lines 1300 to 1560. Also remember to save a copy to tape or disc before running it, in case of any mistakes.

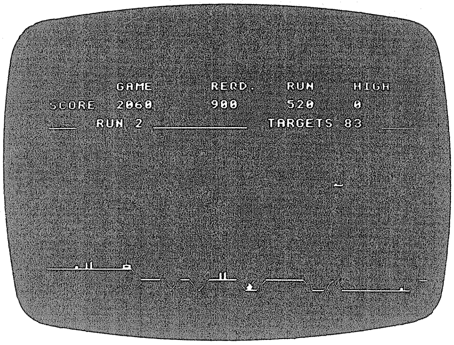
```
   10 REM PROGRAM DIVE BOMBER
   20 REM AUTHOR N.MALLINSON
   30 REM VERSION B1.1
   40 REM BEEBUG JULY 1984
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60:
  100 ON ERROR GOTO 2760
  110 MODE 5
  120 VDU 23,1,0;0;0;0;
  130 PROCtitle
  140 MODE 4
  150 VDU 19,0,4,0,0,0
```



```
Use your aircraft to bomb enemy
installations, and score:
  SUPPLY DUMP                20
  AA GUN                     40
  COMMUNICATIONS CENTRE      80
  COMMAND CENTRE            160
If you reach or surpass the required
score, you will get another run.
Successive runs become harder as the
game speeds up, and the ground may
become higher, with an increase in
the required score.
If you crash or are shot down, the
game ends.
  AIRCRAFT CLIMB    ':'
  AIRCRAFT DIVE     '/'
  RELEASE BOMB      'Z' (when in dive)
Press the Space bar to continue :
```

```
  160 PROCinit
  170 PROCinstructions
  180 PROCnewgame
  190 REPEAT
  200 MODE 4
  210 VDU 19,0,4,0,0,0
  220 VDU 23,1,0;0;0;0;:CLS
  230 PROCnewrun
  240 PRINTTAB(1,7)"PRESS SPACE BAR TO
START"
  250 REPEAT:VDU 7:UNTIL GET=32
  260 PRINTTAB(1,7)"TAKE OFF !";SPC(14)
:VDU 7
  270 REPEAT
  280 IF F%=0 GOTO 320
  290 IF ?(22775+320*F%)<>0 THEN PROCbo
mbhit:GOTO 320
  300 F%=F%+1:IF?(22775+320*F%)<>0 THEN
 PROCbombhit:GOTO 320
  310 PRINTTAB(30,F%-1);SPC1;TAB(30,F%)
;CHR$(227)
  320IF X%=0 THEN L%=225 ELSE X%=X%-1:S
OUND&10,-10,1,5:IF X%=0 THEN PRINTTAB(1
,7);SPC(24):PRINTTAB(30,K%);CHR$(225):G
OTO 510 ELSE GOTO 510
  330 IF INKEY(-73) AND K%>6 THEN L%=22
4:N%=K%-1
  340 IF INKEY(-105) THEN L%=226:N%=K%+1
  350 IF ?(22775+320*N%)<>0 THEN PROCpl
anehit(N%):GOTO 690
  360 PRINTTAB(30,K%);SPC1;TAB(30,N%);C
HR$(L%):K%=N%
  370 SOUND&11,-1,130+40*(L%-224),1:SOU
NDch%,-8,3,5
  380 IF INKEY(-98) AND F%=0 AND L%=226
 THEN F%=K%+1 ELSE GOTO 410
  390 IF ?(22775+320*F%)<>0 THEN PROCbo
mbhit:GOTO 410
  400 PRINTTAB(30,F%);CHR$(227)
  410 IF T%=0 AND U%>0 THEN U%=U%+1
  420 IF U%>30 OR V%<7 THEN PRINTTAB(U%
,V%);SPC1:T%=0:U%=0
  430 IF T%=1 THEN PRINTTAB(U%,V%);SPC1
:IF V%>6 AND U%<31 THEN U%=U%+1:V%=V%-1
:PRINTTAB(U%,V%);CHR$(240)
  440 IF U%>0 AND T%=0 AND 27-U%+I%<=V%
-K% THEN T%=1:PRINTTAB(U%,V%);CHR$(240)
:SOUND ch%,2,6,5
  450 IF G%=232 AND U%=0 AND B%>0 THEN
U%=1:V%=H%-1:I%=RND(5)
  460 IF D%=0 AND B%>0 THEN B%=B%+1
  470 IF B%>30 OR C%<7 THEN PRINTTAB(B%
,C%);SPC1:D%=0:B%=0
  480 IF D%=1 THEN PRINTTAB(B%,C%);SPC1
:IF C%>6 AND B%<31 THEN B%=B%+1:C%=C%-1
:PRINTTAB(B%,C%);CHR$(240)
  490 IF B%>0 AND D%=0 AND 28-B%+E%<=C%
-K%+(1+C%-K%)*(225-L%) THEN D%=1:PRINTT
AB(B%,C%);CHR$(240):SOUNDch%,2,6,5
  500 IF G%=232 AND B%=0 THEN B%=1:C%=H
%-1:E%=RND(3)
  510 A%=0:IF Z%<1 THEN G%=237:Z%=Z%-1:
GOTO 610
  520 IF G%=235 THEN A%=A%-1
  530 IF H%>28 THEN G%=235:GOTO 590
  540 IF H%<Y% THEN G%=234:A%=A%+1:GOTO
 590
  550 R%=RND(127):G%=229
  560 REPEAT:R%=R% DIV 2:G%=G%+1:UNTIL
R%=0
  570 IF G%=235 THEN G%=G%-RND(4) DIV 4
  580 IF G%=234 THEN A%=A%+1
  590 H%=H%+A%
  600 IF G%<234 AND Z%>0 THEN Z%=Z%-1:P
RINTTAB(31,5);Z%;SPC1
  610 CALL SCRLR
  620 PRINTTAB(0,H%);CHR$(G%)
  630 IF X%<>0 GOTO 680
  640 IF ?(22775+320*K%)<>0 THEN PROCpl
anehit(K%):GOTO 690
  650 IF D%=1 THEN PRINTTAB(B%,C%);SPC1
:IF C%>6 AND B%<31 THEN B%=B%+1:C%=C%-1
:PRINTTAB(B%,C%);CHR$(240)
  660 IF T%=1 THEN PRINTTAB(U%,V%);SPC1
:IF V%>6 AND U%<31 THEN U%=U%+1:V%=V%-1
:PRINTTAB(U%,V%);CHR$(240)
  670 IF J%>0 THEN J%=J%-1:IF J%<1 THEN
 ch%=&10
  680 FOR M%=1 TO DL%:NEXT
  690 UNTIL Z%<-40
  700 IF Z%>-50 THEN PROClanding ELSE I
F L%=242 THEN PRINTTAB(1,7)"SHOTDOWN !"
 ELSE PRINTTAB(1,7)"CRASHED !"
  710 TIME=0:REPEAT UNTIL TIME>100
  720 *FX 15,0
  730 UNTIL Z%=-50
  740 PROChighscores:PRINTTAB(3,M%)"ANO
THER GAME (Y/N) ?"
  750 VDU 7:I$=GET$:IF I$="Y" THEN CLS:
GOTO 180
  760 IF I$<>"N" GOTO 750
  770 *FX15,0
```

≫



```
      GAME      RECD.    RUN    HIGH
SCORE 2060      900      520    0
____ RUN 2 __________ TARGETS 83 ____
```

```
  78Ø MODE 7
  79Ø END
  8ØØ:
 1ØØØ DEF PROCbombhit
 1Ø1Ø IF ?(22774+32Ø*F%)=32 THEN PRINTT
AB(3Ø,F%);SPC1:GOTO 1Ø9Ø
 1Ø2Ø IF ?(22774+32Ø*F%)<1ØØ THENSOUND&
1Ø,-1Ø,6,4:J%=1:GOTO 1Ø7Ø
 1Ø3Ø Q%=1Ø*?(22769+32Ø*F%)
 1Ø4Ø W%=W%+Q%:S%=S%+Q%
 1Ø5Ø PRINTTAB(7,3);S%;TAB(25,3);W%
 1Ø6Ø SOUND&1Ø,3,6,4Ø:J%=5
 1Ø7Ø PRINTTAB(3Ø,F%);CHR$(228)
 1Ø8Ø ch%=&11
 1Ø9Ø IF F%-1<>K% THEN PRINTTAB(3Ø,F%-1
);SPC1
 11ØØ F%=Ø
 111Ø ENDPROC
 112Ø:
 113Ø DEF PROClanding
 114Ø IF F%>Ø THEN PRINTTAB(3Ø,F%);SPC1
:SOUND&13,Ø,Ø,1
 115Ø PRINTTAB(1,7)"LANDING !"
 116Ø REPEAT
 117Ø IF K%<H%-1 THEN K%=K%+1:PRINTTAB(
3Ø,K%-1);SPC1;TAB(3Ø,K%);CHR$(226):SOUN
D&1Ø,-8,2,5
 118Ø FOR M%=1 TO 2*DL%:NEXT
 119Ø IF K%=H%-1 THEN PRINTTAB(3Ø,K%);C
HR$(243)
 12ØØ UNTIL K%=H%-1
 121Ø IF W%>=RS% THEN PRINTTAB(1,7)"MIS
SION ACCOMPLISHED !" ELSE PRINTTAB(1,7)
"FAILED REQUIRED SCORE.":Z%=-5Ø
 122Ø S%=S%+1ØØ:PRINTTAB(7,3);S%
 123Ø FOR M%=Ø TO 1ØØØØ:NEXT
 124Ø ENDPROC
 125Ø:
 126Ø DEF PROCinit
 127Ø DIM SET 4ØØ
 128Ø FOR PASS=Ø TO 1
 129Ø P%=SET
 13ØØ [OPT PASS*2
 131Ø .SET    LDA #&38:STA &74:LDA #&72:
STA &75\set start
 132Ø         LDX #39:LDA #Ø\clear base
move store
 133Ø .clear STA SET+3ØØ,X:DEX
 134Ø         BPL clear:RTS\end when all
 done
 135Ø .SCRLR STA SET+3ØØ\store A% move
 136Ø         LDA &74:STA &7Ø:LDA &75:ST
A &71\low base=start
 137Ø         LDX #39\index for 39 squar
es
 138Ø .sqr    LDA #Ø:LDY #7\blank out sq
uare
 139Ø .blank STA(&7Ø),Y:DEY:BPL blank
 14ØØ         LDA SET+3ØØ,X:STA SET+3Ø1,
X\shift move up store
 141Ø         BEQ level:BMI up\detect mo
ve
 142Ø         LDA &7Ø:CLC:ADC #&38:STA &
7Ø\move down
 143Ø         LDA &71:ADC #1:STA &71:CLC
:BCC ind
 144Ø .up     LDA &7Ø:SEC:SBC #&48:STA &
7Ø\move up
 145Ø         LDA &71:SBC #1:STA &71:CLC
:BCC ind
 146Ø .level LDA &7Ø:SEC:SBC #8:STA &7Ø
\back one square
 147Ø         LDA &71:SBC #Ø:STA &71
 148Ø .ind    DEX:BPL sethb:RTS\end if i
ndex neg
 149Ø .sethb LDA &7Ø:CLC:ADC #8:STA &72
\high base=low base+8
 15ØØ         LDA &71:ADC #Ø:STA &73
 151Ø         CPX #38:BNE trans\far righ
t square?
 152Ø         LDA &72:STA &74:LDA &73:ST
A &75:\yes-start=high base
 153Ø .trans LDY #7\transfer data
 154Ø .byte  LDA(&7Ø),Y:STA(&72),Y:DEY:
BPL byte
 155Ø         BMI sqr\go do next square
 156Ø ]
 157Ø NEXT
 158Ø CLS
 159Ø VDU23,224,Ø,8Ø,4Ø,16,8,5,2,Ø:REM
climb plane
 16ØØ VDU23,225,Ø,Ø,Ø,Ø,97,255,Ø,Ø:REM
level plane
 161Ø VDU23,226,4,2,4,4Ø,8Ø,32,64,Ø:REM
 dive plane
 162Ø VDU23,227,Ø,16,8,2Ø,32,64,Ø,Ø:REM
 bomb
 163Ø VDU23,228,Ø,146,84,56,254,56,84,1
46:REM expl
 164Ø VDU23,23Ø,16,16,16,16,254,198,254
,255:REM cmd cen
```

»

```
 1650 VDU23,231,8,8,8,8,8,232,232,255:R
EM cms cen
 1660 VDU23,232,2,4,104,48,120,248,112,
255:REM gun
 1670 VDU23,233,0,2,102,102,102,102,102
,255:REM dump
 1680 VDU23,234,1,2,4,8,16,32,64,128:RE
M gnd down
 1690 VDU23,235,128,64,32,16,8,4,2,1:RE
M gnd up
 1700 VDU23,236,0,0,0,0,0,0,0,255:REM g
nd level
 1710 VDU23,237,255,129,129,129,129,129
,65,129:REM runway
 1720 VDU23,240,0,0,0,0,0,0,32,64:REM s
hot
 1730 VDU23,241,8,8,8,8,8,8,0,0:REM tra
il
 1740 VDU23,242,24,8,8,8,8,24,24,8:REM
hit plane
 1750 VDU23,243,0,0,0,0,0,97,255,72:REM
 taxi plane
 1760 REM * define envelopes *
 1770 ENVELOPE 1,130,5,3,5,50,60,70,0,0
,0,0,0,0
 1780 ENVELOPE 2,128,10,0,0,6,0,0,0,0,0
,0,0,0
 1790 DIM HSC%(6),NAME$(6)
 1800 FOR M%=0 TO 6:HSC%(M%)=0:NAME$(M%
)=STRING$(11,CHR$32)
 1810 ENDPROC
 1820:
 1830 DEF PROCnewgame
 1840 DL%=392:S%=0:Y%=21
 1850 RS%=700:RN%=0
 1860 ENDPROC
 1870:
 1880 DEF PROCnewrun
 1890 FOR M%=0 TO 38
 1900 PRINTTAB(M%,5);CHR$(236);TAB(M%,2
0);CHR$(237)
 1910 NEXT
 1920 CALL SET
 1930 IF RS%<2500 THEN RS%=RS%+100
 1940 IF DL%>2 THEN DL%=DL%-30
 1950 RN%=RN%+1
 1960 IF Y%>10 THEN Y%=Y%-1
 1970 G%=236:H%=20:K%=19:L%=243:N%=K%
 1980 F%=0:B%=0:D%=0:U%=0:T%=0
 1990 J%=0:ch%=&10:W%=0:Z%=100:X%=15
 2000 PRINTTAB(7,1)"GAME";TAB(17,1)"REQ
D.";TAB(25,1)"RUN";TAB(32,1)"HIGH";TAB(
0,3)"SCORE";TAB(7,3);S%;TAB(17,3);RS%;T
AB(25,3);W%;TAB(32,3);HSC%(0)
 2010 PRINTTAB(3,5)"  RUN ";RN%;SPC1;TA
B(21,5)"  TARGETS ";Z%;SPC1
 2020 PRINTTAB(30,K%);CHR$(L%)
 2030 R%=RND(-TIME)
 2040 ENDPROC
 2050:
 2060 DEF PROCplanehit(k%)
 2070 Z%=-50
 2080 IF ?(22775+320*k%)<>64 THEN PRINT
TAB(30,k%+225-L%);SPC1;TAB(30,k%);CHR$(
228):SOUND&10,3,6,40:ENDPROC
 2090 SOUND&11,-1,255,1
 2100 L%=242
 2110 REPEAT
 2120 PRINTTAB(30,k%);CHR$(242);TAB(30,
k%-1);CHR$(241)
 2130 k%=k%+1:pk%=?(22775+320*k%)
 2140 FOR M%=1 TO 1000:NEXT
 2150 UNTIL pk%<>0 AND pk%<>64
 2160 PRINTTAB(30,k%);CHR$(228);TAB(30,
k%-1);CHR$(241)
 2170 SOUND&11,-12,3,-1:SOUND&10,-12,3,
-1
 2180 ENDPROC
 2190:
 2200 DEF PROCinstructions
 2210 CLS
 2220 PRINT'TAB(2)"Use your aircraft to
 bomb enemy";SPC(9);"installations, and
 score:"
 2230 PRINT'TAB(4)"SUPPLY DUMP";SPC(13)
;;CHR$(233);"  20"
 2240 PRINT'TAB(4)"AA GUN";SPC(18);CHR$
(232);"  40"
 2250 PRINT'TAB(4)"COMMUNICATIONS CENTR
E";SPC(3);CHR$(231);"  80"
 2260 PRINT'TAB(4)"COMMAND CENTRE";SPC(
10);CHR$(230);" 160"
 2270 PRINT'TAB(2)"If you reach or surp
ass the required";SPC(4);"score, you wi
ll get another run."
 2280 PRINT'TAB(2)"Successive runs beco
me harder as the";SPC(4);"game speeds u
p, and the ground may";SPC(6);"become h
igher, with an increase in";SPC(6);"the
 required score."
 2290 PRINT'TAB(2)"If you crash or are
shot down, the";SPC(6);"game ends."
 2300 PRINT'TAB(4)"AIRCRAFT CLIMB";SPC(
3);"':'"
 2310 PRINT'TAB(4)"AIRCRAFT DIVE";SPC(4
);"'/'"
 2320 PRINT'TAB(4)"RELEASE BOMB";SPC(5)
;"'Z'  (when in dive)"
 2330 PRINT TAB(2,29);"Press the Space
bar to continue :";
 2340 REPEAT UNTIL GET=32
 2350 ENDPROC
 2360:
 2370 DEF PROCtitle
 2380 COLOUR 1
 2390 FOR M%=6 TO 10 STEP 4
 2400 PRINTTAB(5,M%);STRING$(11,CHR$(61
)):NEXT
 2410 COLOUR 2
 2420 PRINTTAB(5,8);"DIVE-BOMBER"
 2430 COLOUR 1
 2440 PRINTTAB(10,16)"by"
 2450 COLOUR 3
 2460 PRINTTAB(5,18)"N.Mallinson"
 2470 TIME=0:REPEAT UNTIL TIME>200
 2480 ENDPROC
 2490:
 2500 DEF PROCphstable
 2510 CLS
 2520 PRINTTAB(10,4)"HIGH SCORE TABLE"
 2530 PRINTTAB(6,8)"No.";TAB(13,8)"NAME
";TAB(25,8)"SCORE"
 2540 M%=-1
 2550 REPEAT
 2560 M%=M%+1
 2570 IF HSC%(M%)=0 THEN UNTIL TRUE=TRU
E ELSE PRINTTAB(6,11+2*M%);M%+1;TAB(12,
11+2*M%);NAME$(M%);TAB(25,11+2*M%);HSC%
(M%):UNTIL M%=5
 2580 IF HSC%(0)=0 THEN PRINTTAB(12,16)
"NO SCORE YET !"
 2590 IF Z%=0 THEN PRINTTAB(3,27)"INSTR
UCTIONS (Y/N) ?"
 2600 ENDPROC
 2610:
 2620 DEF PROChighscores
 2630 M%=6
 2640 REPEAT
 2650 M%=M%-1
 2660 IF S%>HSC%(M%) THEN HSC%(M%+1)=HS
C%(M%):NAME$(M%+1)=NAME$(M%):UNTIL M%=0
 ELSE M%=M%+1:UNTIL TRUE=TRUE
 2670 IF M%>5 THEN M%=9:ENDPROC
 2680 CLS
 2690 PRINTTAB(1,7)"YOU ARE ON THE HIGH
 SCORE TABLE !"
 2700 PRINTTAB(3,12)"PLEASE INPUT YOUR
NAME"''TAB(5)"(max. 10 letters)"
 2710 REPEAT:VDU7:PRINTTAB(10,17);SPC(1
5):INPUT TAB(10,17)"? "N$:UNTIL LEN(N$)
<11
 2720 HSC%(M%)=S%:NAME$(M%)=N$
 2730 PROCphstable:M%=27
 2740 ENDPROC
 2750:
 2760 ON ERROR OFF:MODE 7
 2770 IF ERR<>17 REPORT:PRINT" at line
";ERL
 2780 END
```

POINTS ARISING

TELETEXT ADAPTOR REVIEW

In the review of the Acorn Teletext Adaptor in BEEBUG Vol.2 No.10 it was stated that the TFS (Teletext Filing System) chip automatically grabbed extra memory even when the adaptor was not switched on. This is true if the adaptor is disconnected from the 1MHz bus, but the chip acts in an intelligent way if the adaptor is connected and only resets PAGE when the adaptor is switched on. However, if you move your Beeb from its normal place of work without the Teletext adaptor you will have to remove the TFS chip as well, or reset PAGE every time you switch the machine on or press Break. Thanks to Dick Orton for pointing this out.

MACHINE CODE GRAPHICS AGAIN

In part 3 of this series in BEEBUG Vol.2 No.10 there are two small errors of detail. At the bottom of page 28 the correct code for 'magenta and cyan' is 54 and not 53, while the data lines 2010, 2020 and 2030, near the bottom of page 29 should contain the same figures. Thus line 2010 should contain 26 and not 28 as printed. You will very probably have realised that the table of colour codes at the bottom of page 28 was for mode 2 and not mode 1 as stated. We apologise for these errors.

TELETEXT MODE SCREEN DUMP

The mode 7 screen dump included on the magazine cassette for April (Vol.2 No.10) assumes that the Epson FX80 printer is not set to produce a Linefeed automatically following receipt of a Carriage return (i.e. *FX6,0 is normally set). If the FX80 is set to do this, then the screen dump produces double line spacing. You can of course change the setting of your printer, or change the '10' in line 1800 to '0' to read:

```
1800 DATA 3,192,1,94,27,0,13
```

SPITFIRE AEROPLANE DISPLAY

We were very disappointed that a hiccup during production resulted in an error occurring in the data for our Spitfire aeroplane. In line 180, the second number should be -50 and not just 50. You may have noticed that several numbers in this area were touched up, unfortunately this one incorrectly.

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.

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

MEMBERSHIP COSTS:
U.K.
£5.40 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
Eire and Europe
Membership £16 for 1 year.
Middle East £19
Americas and Africa £21
Elsewhere £23
Payments in Sterling preferred.

Subscriptions & Software Address
BEEBUG
PO BOX 109
High Wycombe
Bucks

Subscriptions and Software Help Line
St.Albans
(0727) 60263
Manned Mon-Fri
1pm-4pm

PROGRAMS AND ARTICLES

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

HINTS

There are prizes of £5 and £10 for the best hints each month.

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

Editorial Address
BEEBUG
PO Box 50
St Albans
Herts

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

BEEBUG NEW ROM OFFER

1.2 OPERATING SYSTEM

A special arrangement has been agreed between Acorn and BEEBUG whereby BEEBUG members may obtain the 1.2 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 machine 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.

FREE

NEW ROMS FOR OLD
EXCHANGE YOUR 1.0 FOR THE 1.2

We can now exchange your old 1.0 operating system for the new 1.2, free of charge. To take advantage of this offer, please send your 1.0 (supplied on eprom with a carrier board), in good condition to the address below.

£5 FOR YOUR OLD 1.0

If you have the 1.0 operating system and have already bought a 1.2, we will exchange the 1.0 (supplied on eprom with a carrier board) for a £5 voucher. This voucher may be used against any purchase from BEEBUGSOFT.

ADDRESS FOR 1.2 OS:-
ROM Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD. HP10 8HQ.

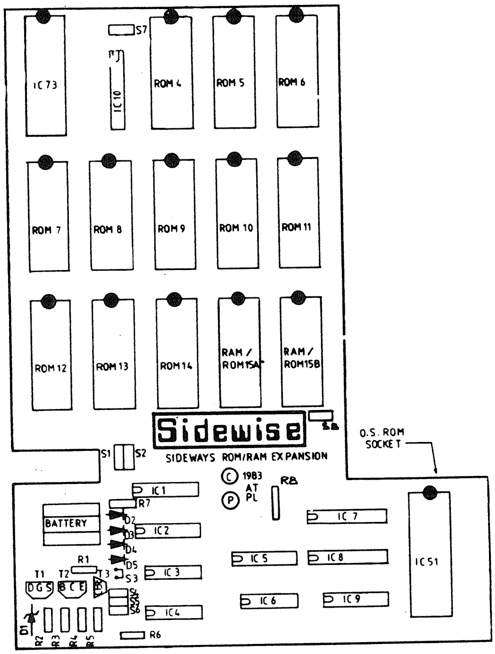
BEEBUGSOFT

ATPL'S SIDEWAYS ROM EXPANSION BOARD

SPECIAL PRICE TO MEMBERS £39.00 inc.

Save £5.70 on normal price of £44.70

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



HOW TO ORDER

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

The address for SIDEWAYS is:
BEEBUGSOFT, PO Box 109, High Wycombe, Bucks. HP10 8HQ.

MAGAZINE CASSETTE OFFER

To save wear and tear on fingers and brain, we offer, 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.

All previous magazine cassettes (from Vol.1 No.10) are available.

This month's cassette (Vol.3 No.3): includes: Bach music program, four demonstartion programs of BEEBUG workshop routines, Multi-screen Slide Show, a Fast Colour Fill routine, a Printer Spooler utility, a program for freezing ans saving screen displays, Snip Snap game and Dive Bomber game.

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:
BEEBUG, PO Box 109, High Wycombe, Bucks, HP10 8HQ.

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 BEEBUG magazine is available. These binders are dark blue in colour with 'BEEBUG' in gold lettering on the spine. They allow you to store the whole of one volume of the magazine as a single reference book. Individual issues may be easily added or removed, thus providing ideal storage for the current volume as well.

BINDER PRICE
U.K. £3.90 inc p&p, and VAT.
Europe £4.90 inc p&p
(no VAT payable).
Elsewhere £5.90 inc p&p
(no VAT payable).

Make cheques payable to BEEBUG.
Send to Binder Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP10 8HQ.
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