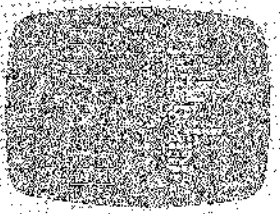
£1.00

BEEBUG

FOR THE BBC MICRO

Vol 2 No 1 APRIL/MAY 1983



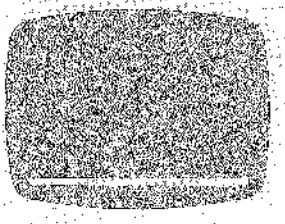
TAPE RECORDERS REVIEWED

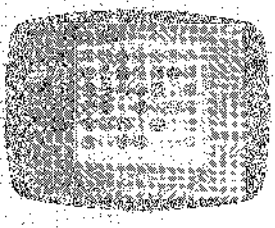
MUSIC COMPOSER PROGRAM

USER GUIDE ERRORS

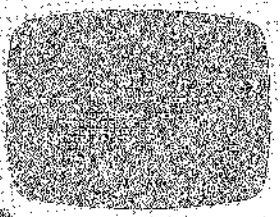
MAGAZINE CASSETTE OFFER

INVASION

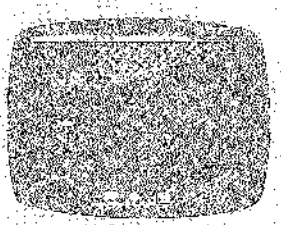




FOUR-IN-A-ROW



BASIC PROGRAM EDITOR



ACORN – SOFT REVIEWS

BAR CHART GENERATOR

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 17,000

EDITORIAL

ANNIVERSARY

We have survived our first year! BEEBUG magazine has now reached volume two, and to mark the passing of time we have brought together some extra items with this issue.

INDEX

Firstly the four centre pages of the magazine contain a referenced index for the ten issues of volume one. We hope that this will make it easier to find things.

BINDER OFFER

Secondly we are offering a binder for the magazine. This is A5 size, in dark blue with 'BEEBUG' printed in gold on the spine. I have been using a sample binder for some time now, and it really makes referencing back issues a great deal easier. Taken with the cumulative index it enables the whole volume to be used as a single reference book. See back page for ordering details.

REFERENCE CARD

Thirdly, this issue is accompanied by a BEEBUG reference card (unless the magazine was ordered as a back issue). This is packed with a wealth of data, which we hope will save hours of searching through the user guide. And in practice the Reference Card goes a little beyond the User Guide, giving some of the latest FX calls. We HOPE that the Card is relatively free from errors - it has been carefully checked - but only time will tell.

MAGAZINE CASSETTE

Fourthly, each month we shall be putting the programs from the magazine on to cassette. These will be available at £3.00 + 50p p&p (inc VAT) to members. The obvious advantage of the cassettes is that they will save some time and anguish in getting programs running. There is also a further advantage: they allow us to provide extra material. For example, in the cassette for this issue, we have included a program produced by the Music Composer in this issue, which plays Beethoven's Bagatelle in b flat.

APOLOGY

We also have an apology to make. Because of space restrictions in this issue (partly caused by dropping the extra small print which we used experimentally in Feb and March), we have had to postpone the promised 40 user key program, and part 3 of Using Files, until next issue.

We must also apologise for calling this issue 'April/May' rather than 'April'. One or two computer shops now stock BEEBUG, and we have been asked if the cover date can be advanced so that issues on their shelves do not appear to be out of date once they have been on display for a few days. Being unable to bring the publication date forward, we are changing the month on the cover. The actual date of publication and the issue number will remain unchanged. The next issue (issue no. 2) will be called the June issue, and will come out towards the end of May.

O.S. 1.2 ROM

We have now supplied around 6000 members with O.S. 1.2, and we expect all outstanding orders to be cleared by 15 April. Because of this we can now accept multiple orders from institutional members. There can be no quantity discount in such cases, and the price of £5.85 per unit stands. Again, you should allow 4 - 6 weeks for delivery.

NOTICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOARD

This month's hint winners are Richard Bean (£10) and Chris Eilbeck (£5).

BEEBUG MAGAZINE

GENERAL FEATURES

Editorial	2
Quick Quiz Best Time	6
Tape Recorders Reviewed	7
Music Composer	10
Postbag	15
Acornsoft Games Review	17
Secrets of Program Saving	18
Brain Teaser	19
Torch Z80 Disc Pack Review	20
Utility Editor	22
Newcomers Start Here - Mode 7	26
Disc Menu Program	28
Double Height Characters in all Modes	30
Points Arising	31
Implementing Pseudo Ops and Macros	32
Bar Chart Generator (32k)	34
Mini Text El Update	38
User Guide Errors	40
Software Update	41
Magazine Cassette Offer	44
Binder Offer	44
Index to Volume 1 -	centre pages

PROGRAMS

Four In A Row (32k)	4
Composer (16k)	13
Utility Editor (16k/32k)	24
Disc Menu (32k)	28
Double Height Characters (16k/32k)	30
Bar Chart (32k)	36
Invasion (16k)	39

HINTS, TIPS & INFO

Key Boot (Discs)	6
Time Delays	9
Faster Circles and Regular Polygons	21
Space Invader Prompt	21
Removing Spaces	25
TV Shake Fix	29
Programmable Function Calls	29
Scroll Commands	33
Illegal PRINT TAB	38
ON ERROR Off	40
Memory Saving on Arrays	41

Program tested on O.S. 0.1 and 1.2

FOUR IN A ROW (32k)

Program by John Webb

THE GAME

Four-In-A-Row is a nice colourful implementation of the commercially available game 'Connect Four'. The object is to get four counters in a row, either horizontally, vertically or diagonally on a 6 x 7 playing board. It is a game where two players take alternate turns, deciding in which column to place a counter. You cannot choose what row to put it in, counters just pile up from the bottom.

In this implementation you can play against the Beeb, or another person. Four In A Row is a game of strategy, and the program plays quite a fair game. Generally speaking it will win unless you can set a trap for it by gradually building up two interrelated rows of four. This implementation plays a better game than the only other computer version that I have seen, which ran on a UK101.

OPERATING NOTES

The game uses Mode 1, and therefore requires a 32k machine, but it will fit into a disc machine if you set PAGE to &1100 (ie. enter PAGE=&1100 <return>) before loading (or entering from keyboard) - but remember not to press Break. Once you have it on disc it may be useful to have a short header program saved separately on disc which resets PAGE. For example:

```
10 REM short header for 4INAROW
20 PAGE=&1100
30 CHAIN"4"
```

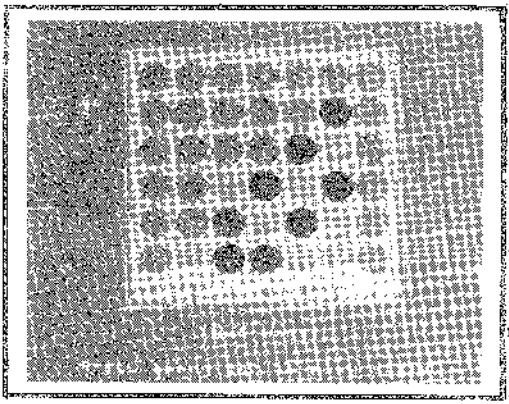
Save this under the title "4INAROW", and save the main program under the title "4". Then to load just CHAIN "4INAROW"

The colours used in the program are pretty, but the yellow is a little difficult to see against the surrounding white frame. To make it magenta, add:

```
145 VDU19,2,5,0,0,0
```

Alternatively, if watching in black & white, try deleting line 280, or better still add:

```
145 VDU19,2,10,0,0,0
```



```
10REM**Four-in-a-Row by John Webb**
20DIMposition%(6,7):PROCcircles
30ENVELOPE1,6,0,0,0,0,0,0,0,121,-10,-5,-2,1
20,120
40ENVELOPE2,6,4,-8,-4,16,16,32,64,64,-64,
-64,128,0
50MODE7:PROCtitles
60go%=1:win%=0:turn%=0:comp%=0
70FORrow%=1TO6
80FORcol%=1TO7
90position%(row%,col%)=0
100NEXT:NEXT
110MODE1
120PROCsetupgame
130REPEAT
140PROCturn
150IF go%=0 AND comp%=0 PROCcolour
160IF go%=0 AND comp%=1 PROCcomputer:PROCc
olour
170IF go%=1 PROCcolour
180PRINTTAB(0,8)"Last go"'"col: ";column%
190SOUND&11,1,86,111:screen%=14347+48*colu
mn%:REPEAT:screen%=screen%+1920:UNTIL?screen
%>0
200missweigh%=0:PROCweighandcheck:missweig
h%=1
210UNTILwin%=99ORturn%=42:PROCwin:GOTO60
220REM*****************************
230REM*****************************
240DEFPROCsetupgame
250REM This draws the board, and establish
es who is playing.
```

```
  260VDU19,3,0;0;:MOVE285,895:DRAW285,205:DR
AW1020,205:DRAW1020,895:DRAW285,895
  270MOVE300,880:MOVE300,220:PLOT85,1005,880
:PLOT85,1005,220:VDU20
  280VDU19,0,4,0,0,0
  290FORR=1TO6:FORC=1TO7:PRINTTAB(7+3*C,2+3*
R);C$;:NEXT:NEXT
  300VDU5:MOVE355,260:GCOL0,0:PRINT"1  2  3
 4  5  6  7":VDU4
  310VDU23;8202;0;0;0;
  320REPEAT:PROCclear:PRINTTAB(11,27)"Do you
 want to play":PRINTTAB(10,29)"the computer?
(Y or N) ":ans$=GET$:UNTILans$="Y"ORans$="N"
  330IF ans$="N"THEN ENDPROC
  340REPEAT:PROCclear:comp%=1:PRINTTAB(12,27
)"Do you want to go":PRINTTAB(13,29)"first?(
Y or N) ":ans$=GET$:UNTILans$="Y"ORans$="N"
  350IF ans$="Y" go%=0
  360ENDPROC
  370REM*****************************
  380REM*****************************
  390DEFPROCcolour
  400REM This colours display at next availa
ble position
  410col%=column%:row%=0:REPEAT:row%=row%+1:
UNTILposition%(row%,column%)=0
  420position%(row%,column%)=1
  430COLOUR(130-go%):PRINTTAB(7+3*column%,23
-3*row%);C$:COLOUR128
  440ENDPROC
  450REM*****************************
  460REM*****************************
  470DEFPROCcircles:REM Draws one circle
  480VDU23,224,255,255,255,252,240,224,224,1
92:VDU23,227,192,128,128,128,128,128,128,192
:VDU23,229,3,1,1,1,1,1,1,3:VDU23,226,255,255
,255,63,31,15,7,3
  490VDU23,232,3,7,15,31,63,255,255,255:VDU2
3,228,0,0,0,0,0,0,0,0:VDU23,225,255,255,0,0,
0,0,0,0:VDU23,231,0,0,0,0,0,0,255,255
  500VDU23,230,192,224,224,240,252,255,255,2
55
  510C1$=CHR$224+CHR$225+CHR$226:C2$=CHR$227
+CHR$228+CHR$229:C3$=CHR$230+CHR$231+CHR$232
:COLOUR3:COLOUR127
  520C$=C1$+CHR$8+CHR$8+CHR$8+CHR$10+C2$+CHR
$8+CHR$8+CHR$8+CHR$10+C3$
  530ENDPROC
  540REM*****************************
  550REM*****************************
  560DEFPROCtitles:VDU23;8202;0;0;0;:REM Can
cels curser
  570PRINTTAB(11,10)CHR$131CHR$141"FOUR IN A
 ROW"
  580PRINTTAB(11,11)CHR$131CHR$141"FOUR IN A
 ROW"
  590wait=INKEY(300)
  600CLS:PRINTTAB(2,5)CHR$130"The object is
to get four of your own":PRINTCHR$130"colour
 in a line in any direction."
  610PRINT"TAB(2)CHR$130"You can choose to
 play against the":PRINTCHR$130"computer or
 against another player. The":PRINTCHR$130"m
achine always plays yellow."
  620PRINTTAB(6,19)CHR$132"Press space bar t
o play."
  630REPEATUNTILGET$=" "
  640ENDPROC
  650REM*****************************
  660REM*****************************
  670DEFPROCwin:PROCclear:SOUND&11,2,100,121
  680REM This responds to a win or draw
  690IF turn%=42 AND win%<>99 PRINTTAB(12,27
)"Honourable  draw!":GOTO720
  700IF go%=0 PRINTTAB(14,27)"Yellow wins!"E
LSE PRINTTAB(16,27)"Red wins!"
  710*FX15,1
  720PRINTTAB(5,30)"Press space bar to play
again.":IF GET$=" "THEN730ELSECLS:END
  730ENDPROC
  740REM*****************************
  750REM*****************************
  760DEFPROCclear:PRINTTAB(0,27);SPC(120)
  770ENDPROC
  780REM*****************************
  790REM*****************************
  800DEFPROCturn
  810REM This displays whose turn next, and
accepts response
  820turn%=turn%+1
  830IF comp%=1AND go%=1THEN910
  840REPEAT:PROCclear:IF go%=0PRINTTAB(8,27)
"Reds turn - ";
  850IF go%=1PRINTTAB(6,27)"Yellows turn - "
;
  860*FX15,0
  870PRINT"which column?":column%=VALGET$:UN
TILcolumn%>0ANDcolumn%<8
  880IFposition%(6,column%)>0THEN840
  890IF go%=1 go%=0 ELSE go%=1
  900ENDPROC
  910PROCclear:PRINTTAB(16,27)"Computing"
  920go%=0
  930ENDPROC
  940REM*****************************
  950REM*****************************
  960DEFPROCcomputer
  970REMThis must calculate a value for colu
mn%
  980flag%=0:IF turn%<4 column%=4:TIME=0:REP
EATUNTILTIME>100:ENDPROC
  990dontgo1%=0:dontgo2%=0:dontgo3%=0:dontgo
4%=0
 1000swap%=0:column1%=0:column2%=0
 1010FORcol%=1TO7
 1020IFposition%(6,col%)>0THEN1100
 1030IF col%=dontgo1% OR col%=dontgo2% OR co
l%=dontgo3% OR col%=dontgo4% THEN1100
 1040screen%=14347+48*col%:REPEAT:screen%=sc
reen%+192:UNTIL?(screen%+192)>0ORscreen%>2
6203
 1050IFscreen%>26203screen%=25867+48*col%
 1060PROCweighandcheck
 1070IFweight%>swap%THENcolumn%=col%
 1080IFweight%>swap%THENswap%=weight%
 1090IFcheck%=3 col%=7
 1100NEXT
 1110IF column1%>0THENcolumn%=column1%:ENDPR
OC
 1120IF column2%>0THENcolumn%=column2%:ENDPROC
 1130flag%=flag%+1
 1140IF turn%<9 OR position%(5,column%)=1 TH
ENENDPROC
 1150screen%=14347+48*column%:col%=column%
 1160REPEAT:screen%=screen%+192:UNTIL?(scre
en%+3840)>0ORscreen%>=22363
```

```
1170PROCweighandcheck
1180IF flag%>4 THENIF dontgo1%>0 AND dontgo
1%<>dontgo3% AND dontgo1%<>dontgo4% THENcolu
mn%=dontgo1%:ENDPROC
1190IF flag%>4 AND dontgo2%>0 THENcolumn%=d
ontgo2%:ENDPROC
1200IF flag%>4THENcolumn%=dontgo3%:ENDPROC
1210IF dontgo1%=column% OR dontgo2%=column%
 OR dontgo3%=column% OR dontgo4%=column% THE
N1000
1220ENDPROC
1230REM*****************************
1240REM*****************************
1250DEFPROCweighandcheck
1260REM This gives a weighting to each poss
ible move selected in PROCcomputer, and also
 checks for a winning line after each turn.
1270weight%=0
1280FORA%=0TO3
1290B%=A%=0AND1ORA%>0AND38+A%
1300FORC%=0TO3:check%=0
1310FORD%=0TO3
1320peek%=?(screen%+B%*48*(D%-C%))
1330check%=peek%=240ANDcheck%+10Rpeek%=15AN
Dcheck%-10Rpeek%=0ANDcheck%
1340NEXT
1350IFmissweigh%=0 THEN1410
1360weight%=weight%-(75+RND(30))*(check%=-2
)-(15+RND(10))*(check%=-1)-(15+RND(10))*(che
ck%=1)-(75+RND(30))*(check%=2)
1370IF check%=3 THENcolumn1%=col%:IF dontgo
1%=0 THENdontgo1%=col%
1380IF check%=3 THENdontgo2%=col%
1390IF check%=-3 THENcolumn2%=col%:IF dontg
o3%=0 THENdontgo3%=col%
1400IF check%=-3 THENdontgo4%=col%
1410IF check%=4 OR check%=-4 win%=99
1420NEXT:NEXT
1430ENDPROC
```

QUICK QUIZ BEST TIME

You may remember in the February issue on page 7 we had a quick quiz. It was what numbers between zero and 999 have the special property that they are the sum of the cubes of their digits? We asked for the FASTEST BASIC program to achieve this.

The fastest that I could achieve was 2.98 seconds (without scrolling), and this includes the two statements necessary for timing (ie TIME=0 and PRINT TIME). If you feel that you have significantly bettered this, please send your program to the editorial address. Check that your program actually does give a better time than when running the program below. We cannot, I'm afraid, respond to these programs but will publish the time and your name.

```
 5TIME=0:A%=100:T%=10
10FORA%=0TO9:B%=A%*A%*A%:F%=B%*A%
20FORB%=0TO9:D%=B%*B%*B%:G%=F%+B%
30FORC%=0TO9
40IFD%+E%+C%*C%*C%=B%+C%+C%PRINT;
    A%;B%;C%
50NEXT,:PRINTTIME
```

Sheridan Williams

DISC HINTS DISC HINTS DISC HINTS DISC HINTS DISC H

KEY BOOT

Richard Bean suggests an economic use for the !BOOT mechanism: There are only 31 files available for use on the BBC DFS, so using !BOOT to merely chain another file is expensive in catalogue space. If, like me, the first thing you do is to set up your soft keys, make that program !BOOT. Even better, run your Basic key-set program, then *SAVE B00 BFF as !BOOT. You can then transparently load your keys at any time by pressing BREAK-SHIFT (set the disc to *OPT4,1 - the *LOAD option). Provided that you program KEY!0 (BREAK) with OLD|M, this causes no problems. (Though if you load the keys into an otherwise 'empty' machine you will get a 'Bad Program' error if you press Key 10. Just type NEW to clear). See BEEBUG vol.1 no.9 p.19 for further details on !BOOT.

TAPE RECORDERS FOR THE BBC MICRO

by Adrian Calcraft

We have received a considerable amount of mail on the subject of cassette recorders for use with the BBC Micro. To help members make their own decisions about which tape recorder is most suitable for their own circumstances, we have looked at four machines.

MACHINES REVIEWED			
1)	BOOTS	CR325	PRICE £23.95
2)	DIXONS	PRINZ TR15	PRICE £17.99
3)	W.H.SMITHS	CCR800	PRICE £29.95
4)	FERGUSON	3T07	PRICE £28.50

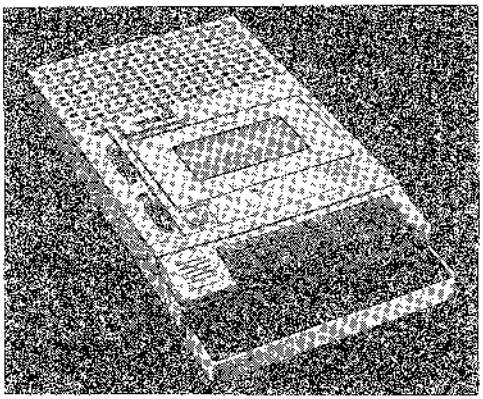
When deciding which machines to look at, we attempted to choose those with sufficient distributors to enable easy purchase for members throughout the country. These machines were all suggested as suitable for use with the BBC Micro.

All the tape recorders are of a similar shape and overall design, being flat and oblong with six control buttons at the front. The average size is about 10 inches long, 6 inches wide and 2 inches high. They all have a flip up lid on top, into which the cassette is inserted.

Other common factors include automatic stop/turn off when reaching the end of a cassette during playback or record, but not during fast forward or rewind. The machines can obviously be used in the normal way without the computer, and as such include a built-in microphone and volume control.

All tests on the tape recorders were made using the standard 'Welcome' cassette, supplied by the BBC. Most cassette recorders are capable of recording and playback at two different levels, through different sockets and leads. The DIN input/output socket is just over 1 cm in diameter and has usually three or five pins. The jack sockets are 3.5 mm in size and will be in a pair on the recorder, one for input, one for output. These two are accompanied, on the machines reviewed here, by a third and smaller (2.5 mm) socket. This provides motor control from the computer.

1) BOOTS CR325

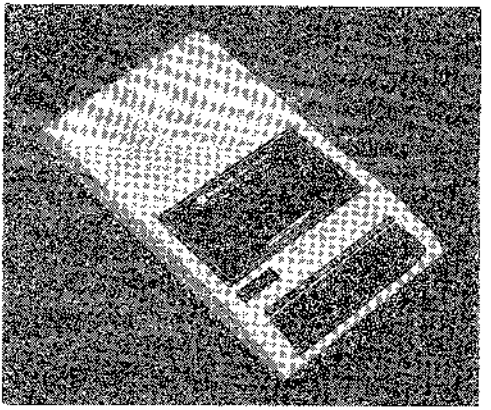


The overall appearance of this machine is very good. It has a tape counter which is easily read, and sliding volume and tone controls, all easily accessible on the top. The controls incorporate a single button for stop/eject and also include a pause button, which can be very useful when using it with the computer.

Both DIN and 3.5 mm jack sockets are available for record/playback, these are located at the front of the machine, below the control buttons. Positioned here they seemed a little in the way, most machines having them to the side.

PERFORMANCE: The machine recorded and played back without problem, through both DIN and jack sockets.

2) DIXONS PRINZ TR15



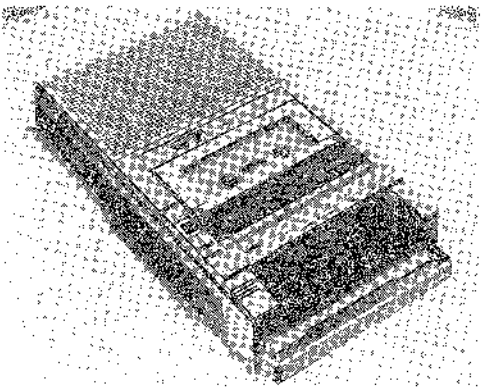
This machine is slightly slimmer than the Boots but gives a less pleasing, plastic appearance. (It is important to remember that this machine is considerably cheaper than all the others.) The TR15 has no tape counter, an omission which could prove inconvenient on many occasions. There is no tone control and the volume control is located on the left side, along with the input/output sockets. In such a position it is hard to tell which way increases and which way decreases the volume. Only 3.5 mm input/output sockets are provided, not a DIN.

PERFORMANCE: The recorder worked well for playback of the 'Welcome' casstte, but proved to be unreliable for saving programs. The problem would appear to be associated with the auto-record level.

Unfortunately while using our demonstration model, the "record protection" feature malfunctioned. This is a small lever which only allows recordings to be made on cassettes which have the rear tabs in place. This lever jammed on the TR15 resulting in accidental over-recording of the Welcome cassette with test data.

PRINZ TR30: As a postscript to this review Dixons have informed us that they will shortly be selling a new recorder, the TR30. It is understood that this machine, which will sell for £21.99 will be more compatible with BBC micro.

3) W.H.SMITHS CCR800



This recorder is sold by Smiths to accompany, dare I say it, their range of Sinclair computers. This tape recorder is very slim and has an impressive appearance. A tape counter is provided and is easily read, being located on the top of the machine.

The volume control is located on the left side of the machine, and is not accompanied by any tone controls. The CCR800 features a 'computer mode' switch, which is recommended for use when saving from the computer. This fixes the automatic recording level control to an 'optimum' level.

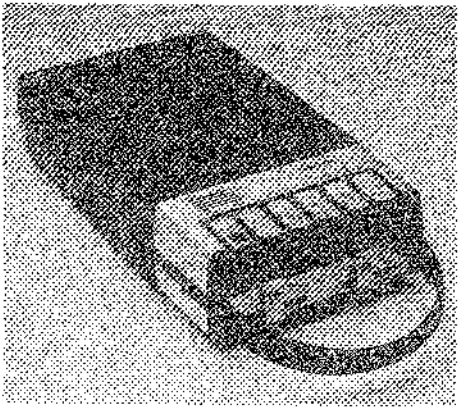
This machine also offers another very useful and unusual feature; the ability to 'cue' and 'review'. This means that if the rewind button is pressed while the play button is locked, the tape will quickly rewind, without resetting the play button, ie. as soon as the rewind button is released, play immediately continues again. The same applies to the fast-forward control. This really is very handy and allows very quick location of, say, the start of a certain program. Unfortunately the

machine does not also offer a pause button.

Both DIN and 3.5 mm jack sockets are provided, being located on the left side of the machine.

PERFORMANCE: When using the DIN socket, the machine recorded and replayed very well with no problems at all. Using the jack socket for recording was slightly less reliable.

4) FERGUSON 3T07



This is the machine which is recommended and sold by Acorn to accompany the BBC micro. The appearance of this machine is good. It is slightly larger than the others reviewed, and so it is suprising that the tape counter, located on the top, is so small as to be hard to read. The controls have a very positive feel, and include a pause control but not the cue/review feature of the Smiths recorder. Volume and tone controls are of the sliding type, being located on the front face of the machine, where their settings are easily read or adjusted.

The standard DIN and 3.5 mm jacks are located on the left side.

PERFORMANCE: The recorder worked well and without fault. Loading and saving of cassettes through both DIN and jack sockets was without problem. However it should be noted that we have received considerable mail from 3T07 owners who have had recording troubles. In past magazines we have published a number of fixes for this condition (eg. see BEEBUG vol.1 no.4 p.29 and BEEBUG vol.1 no.9 p.26).

AVAILABILITY: It is pertinent to mention with this article that we have received letters from unhappy members who were still waiting for delivery of their cassette recorders, ordered through Vector, after very lengthy periods of time. Vector were waiting for delivery themselves and were apparently advising customers that it might be in their best interest, to cancel their order and request a refund. Vector are no longer supplying it as they are now selling a Data Cassette Recorder. This is a machine which is designed specifically for use with computers and consequently cannot be put to any other use. Unfortunately Vector were out of stock of this item when we were collecting machines for review.

CONCLUSION

The Boots recorder offers good reliability at a reasonable price. The Smiths and Ferguson recorders are both somewhat more expensive, and for the extra money you get a more solid and better finished machine, and in the case of the Smiths', a very useful cue-review facility. In view of problems which members have had with the Ferguson, we rate the Boots and the W.H. Smiths as the best buys for their respective prices.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TIME DELAYS

Seeing the two delay PROCs in 'BEEBMAZE' reminded Peter Mabey that a useful tip for setting a delay without disturbing the times is to use the following:

```
1000 DEF PROCdelay(secs)
1010 LOCAL time
1020 time=TIME
1030 REPEAT UNTIL TIME >=time+100*secs
1040 ENDPROC
```

MUSIC COMPOSER (16k)

Program tested on O.S. 1.1 and 1.2

by J. C. Fenton
& Colin Opie

Compose a tune in one, two or three part harmony with J C Fenton's COMPOSER. Single notes or chords may be entered, played back, transposed, and saved to tape with this extremely useful utility.

p dolce

In the Feb '83 issue of BEEBUG (no.9 p.26) J.C Fenton supplied a single line program which turned the QWERTY keyboard into a piano style keyboard, enabling you to play simple tunes. Well from little things do bigger things grow and this month we give a very useful utility, originating from the same author, which enables musical scores to be entered or composed and then stored away on tape or disc. Facilities provided include:

a) Score editing for single notes or chords (up to three voices).
b) Volume and Tempo adjustment.
c) Transposition in steps of a semi-tone.
d) Play facility on single notes, chords or the whole tune.
e) BBC Basic SOUND listing of tune, (with the option of *SPOOLing).
f) Saving and loading of tune.

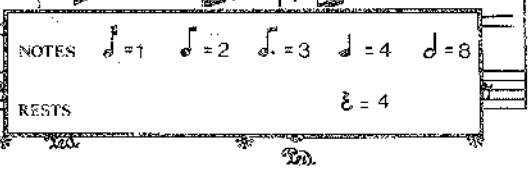
USING THE COMPOSER

Although it's not as simple as the single line program mentioned earlier it is a very easy editor to use. The program is more-or-less self-documenting, and constantly displays a list of single-key commands alongside a 'stave' used for editing purposes.

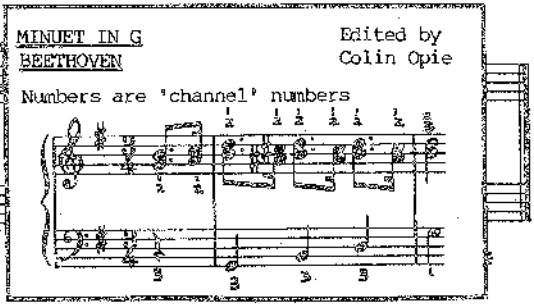
Once you have used the editor for a little while the commands become second-nature and considerable speed can be built up in actually entering a tune. For starting off however it might be useful to look at a description for each command, as given in the table below. Try running the program, and use the commands as specified to enter and play a simple string of notes. The crucial thing to remember is that a note is recorded on the system by entering a number between '1' and '9' specifying note duration.

AN EDITING SESSION

To show how flexible this editor is we will describe how you might enter the few bars of Beethoven's 'Minuet in G', shown here. Before starting you need to decide on the duration for each type of note, bearing in mind the values 1-9 can be used to record a note. In our example we could use:



Three voices exist so three channels will be required. Each voice has to be entered in the same order for each chord, though the order can change for any sequence of notes which start and finish together. This is extremely useful as it means that we can for example adopt the order shown, and hence generate the proper sound in the third bar:



To enter the score first of all set the mode to 'sharps' by typing the command 'M' followed by '1' (this is necessary because there are sharps

in the score). Now start entering the notes/chords. Obtain a 'B' by using the 'note up/down' commands ('>' and '<') and then record the note by using the value '3'. Issue the chord command 'W' and then move to the appropriate 'G' note. Record this with the value '3' also. Now we need a crotchet-rest so again issue the chord command 'W', followed by the interval command 'I', followed by the value '4'.

Repeat a similar process for the first two bars but remember the change in channel order when you get to bar 3. You can enter the 'P' (play) command at any time during editing in order to hear what you have done up to that point. The tempo may need adjusting if the tune doesn't get played at the right speed.

HINTS ON USE

The following points may also be of use:

i) A tune can only be loaded from tape or disc when the program is first entered after typing RUN. You will see from the listing that the filename for the 'music-data' file is always 'TUNE', and the filename for the 'SPOOLed' SOUND statements listing is always 'SPTUNE'.
On a disc system you must rename these files before saving a second version. Owing to the way in which the Disc Filing System extends files it is always best to use a separate disc with the file 'TUNE' only on it for the duration of an editing session. If this is adhered to, the tune can be saved as many times as desired without an 'extend' error occuring. Alternatively the COMPOSE program could be changed so as to allow a filename to be specified by yourself.

ii) For the single key commands to be recognised the keyboard should be in CAPS LOCK mode. It is only necessary to press the key displaying the appropriate character of a command, so you should never use SHIFT even if the position of the character on the key-top would normally imply this.

iii) When editing a tune in more than one 'voice' it is essential to record all 'rests' (ie. gaps in the music) that exist in each of the voices, by using the 'I' (interval) command. If this is not done the individual notes making up a chord (ie. a set of voices) will not be sounded together.

iv) At times you may find that the note indicator goes to the top of the screen and no note is displayed. This simply means that either no note has been allocated to the location or the location holds a 'rest' value. It will occur therefore if you go forward 'F' past the end of the tune entered so far.

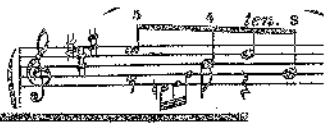
v) Once a tune has been entered the editing facilities supplied can only cope with changing notes or chords which are wrong. It is not possible to go back a few bars and then enter an extra note. It is best to check frequently that what has been entered is correct, or alternatively you could at desired places skip over a number of storage locations by using the 'F' (forward) command, hence leaving a space.

vi) The location counter in the top left-hand corner is a 'note' counter only. Chords will take up more than one location and going backwards and forwards ('B' and 'F') in the music score will cause the counter to change by more than one unit in such cases.

PROGRAM NOTES

1) The "_" character in lines 160 to 190 is the underscore character found on the same key as the 'pound' sign (next to the cursor control keys on the Beeb keyboard); it is not a 'minus' sign.

2) Do not remove spaces from the listing, especially on lines which contain the word 'IF'.

3) The program is currently set to allow up to around 250 notes to be stored (given by variable SZ% in line 60). This enables the program

to run in a model 'A' Beeb. The value of SZ% can be increased by changing the value in line 60 to suit the memory limits of your machine. Clearly a model 'B' computer could store many more than 250 notes.

CREATING Basic MUSIC PROGRAMS

When a listing of the corresponding SOUND statements is requested (by using the 'L' command), you will be given the option to SPOOL the output. If you accept the option (by typing 'Y') then a *SPOOL file, with the name 'SPTUNE', will be created on disc or cassette. This file can later be loaded into Basic for use on its own (ie. without the editor) by typing:

*EXEC SPTUNE

(if the use of SPOOLed files and *EXEC is new to you then see BEEBUG vol.1 no.9 p.20, BEEBUG vol.1 no.10 p.27, and the User Guide p.417).

To play the tune simply type 'RUN', as you would for any Basic program. You can edit the program as you wish, eg. by renumbering and then entering a header, or by appending your latest space-invader program. On this month's cassette tape of all the programs (see elsewhere in this issue) there is a sample Basic music program called 'BEEBAG', created by this music editor, which plays an edited (three voice) version of Beethoven's 'Bagatelle in B-flat, Op119, no.11'.

COMMAND DESCRIPTIONS

'>' and '<' These move the note indicator up and down the stave respectively, so that you can select a particular note.

space Pressing the space-bar will cause the currently indicated note or chord to be sounded.

1 to 9 Once you have selected a note using the '<' and '>' commands the note can be entered into the editor's memory (called 'recording') by using a command from '1' to '9'. The digit corresponds to the desired duration of the note, '1' being the shortest duration.

'&' Signifies 'with' and must be issued between the recording/entering of two or more notes which are to coincide when played, as in the case of a chord. A maximum of three notes may be 'joined' in this way. The command may be cancelled by issuing the 'C' (chord cancel) command.

'P' The entered tune, up to the note-count shown in the top left-hand corner of the screen, is played.

'B' and 'F' When editing an entered tune, eg. when a note is wrong, these commands can be used to set the note-count back or forward a note/chord. This enables any part of the tune recorded so far to be changed. NOTE: a chord must be DELETED (see under 'D') before being re-entered. A single note can simply be 'over-written'.

'V' Sets the volume for when a note is sounded, or a tune played. A prompt is given in the top right-hand corner of the screen and a value from 1 to 9 may be entered, value 9 being the loudest level.

'T' Sets the tempo (ie. speed) at which the tune will be played. All note/chord lengths will be modified as appropriate. A prompt is given in the top right-hand corner of the screen and a value from 1 to 9 may be entered, value 9 being the fastest tempo.

'M' When entering notes/chords the note indicator will normally only display 'straight' notes (ie. C, D, E, ...). If the music contains a number of 'sharps' (#) or 'flats' (b) then the editing 'mode' needs changing. This will then cause either sharps or flats to also be displayed by the indicator (eg. C, C#, D, D#, ...). The prompt for the mode change is given in the top right-hand corner of the screen and a value of 1, 2, or 3 can be entered to obtain sharps, flats, or neither respectively.

'I' Intervals or 'rests' must be entered as well as notes and chords (see later under additional points). This command must be issued before using a command '1' to '9' (see above) to record/enter the length of something. If a 'rest' is entered as part of a chord of 'rests' plus 'notes' it may be

found that the 'ch' (chord) indicator remains in the top left-hand corner of the screen. If this setting is no longer required it can be cancelled simply by giving the 'C' (chord cancel) command.

+, -, 0 These commands enable a complete transposition (ie. shift in pitch) of the whole tune, in steps of half a tone (semi-tone). For example, a 'C' would become a 'C#' if you transposed the music up a semi-tone. The '+' command transposes up, the '-' command transposes down, and the '0' command cancels any transposition and returns the music to its original setting.

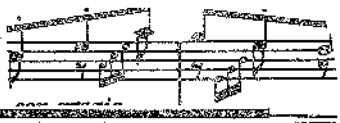
D Deletes a note/chord. A chord or set of notes that have been assigned to be played together, by use of the 'W' command, must be deleted before any part of it is changed, (ie. all parts must be re-entered).

S Saves a data file containing the entered tune on cassette or disc. The filename is always "TUNE" and care must therefore be exercised when using a disc system so as not to overwrite a tune already stored.

L This command will list, in 'paging' mode, the Basic SOUND commands which would produce the entered tune. These could therefore be used within a different program to add music or whatever to it.

E Exit the program. Make sure you have saved ('S') your tune before entering this command.

```
  10 REM MUSIC COMPOSER
  20 REM by J C Fenton and BEEBUG
  30 REM
  40 MODE7:*FX4,1
  50 VDU23;11,0;0;0;0;:ONERRORGOTO580
  60 SZ%=250
  70 C%=0:R%=0:D%=0:A%=0:W%=0:T%=1:S%
=3:V%=-10:E%=5
  80 DIMP%(41,4),N%(SZ%,4),N$(41,2)
  90 FORX=1TO4:FORY=1TO41:READP%(Y,X)
:NEXTY,X
 100 FORX=1TO2:FORY=1TO41:READN$(Y,X)
:NEXTY,X
 110 VDU30:N$(0,1)="n":N$(0,2)="n"
 120 PRINTTAB(1,1);"Would you like to
 load a tune (Y/N)";
 130 X=GET:IFINSTR("Yy",CHR$(X))>0CLS
:PROCLOAD
 140 CLS:L%=N%(C%,4)
 150 PROCVWRITE(0,14,"Music"):PROCVWR
ITE(2,14,"Composer")
 160 PRINTTAB(0,1);CHR$129;C%;CHR$135
;TAB(12,1);"   "
 170 FORY=3TO1STEP2:PRINTTAB(10,Y);"
      ":NEXT
 180 PRINTTAB(12,13);"  ":PROCNOTE(L%)
 190 FORY=15TO23STEP2:PRINTTAB(10,Y);
"      ";:NEXT
 200 PRINTTAB(20,6);"1-9 : Record";TA
B(25,7);CHR$130;"Note length";TAB(22,8
);"W : Chord";CHR$130;"C=canc";TAB(22,
9);"P : Play";TAB(22,10);"B : Back one
";TAB(22,11);"F : Forward one";TAB(22,
12);"V : Volume";TAB(22,13);"T : Tempo "
 210 PRINTTAB(22,14);"M : Mode";CHR$1
30;"#b-";TAB(22,15);"I : Interval";TAB
(20,16);"+0- : Transpose";TAB(22,17);"
D : Delete";TAB(22,18);"S : Save tune"
;TAB(22,19);"L : List";TAB(22,20);"E :End"
 220 PRINTTAB(22,21);"< : Down";TAB(2
2,22);"> : Up";TAB(21,23);CHR$131;"Spa
ce-bar : Sound";
 230 *FX15,1
 240 I%=H%:H%=INSTR("123456789., WBFP
VTMSEILD;-0C",GET$):IFH%=0THEN230
 250 IFH%<10 PROCREC:R%=0:PRINTTAB(1,
2);"  ";:GOTO230
 260 ONH%-9 GOTO270,300,330,340,350,3
80,420,440,450,460,480,580,490,520,530
,470,470,470,570
 270 PROCDEL:L%=L%-1:IF(L%<1) L%=1
 280 IFS%=3THENIFP%(L%,3)=1L%=L%-1:IF
(L%<1) L%=1
 290 GOTO320
 300 PROCDEL:L%=L%+1:IF(L%>41) L%=41
 310 IFS%=3THENIFP%(L%,3)=1L%=L%+1:IF
(L%>41) L%=41
 320 PROCNOTE(L%):GOTO230
 330 PROCSOUND:GOTO230
 340 PROCCHORD:GOTO230
 350 REPEATC%=C%-1:IFC%<1C%=0
 360 UNTILN%(C%,1)MOD&100<2
 370 GOTO400
 380 REPEATC%=C%+1:IFC%>SZ%-3 THEN C%
=SZ%-3
 390 UNTILN%(C%,1)MOD&100<2
 400 PRINTTAB(0,1);CHR$129;C%;CHR$135
;" ";:L%=N%(C%,4):PROCDEL:PROCNOTE(L%)
:IFN%(C%,1)>2PROCNOTE(N%(C%+1,4)):IFN%
(C%,1)=&20:PROCNOTE(N%(C%+2,4))
 410 R%=0:D%=0:GOTO230
 420 *FX15,0
 430 FORX%=0TOC%:PROCPLAY(X%):NEXT:GO
TO230
 440 PROCVOL:GOTO230
 450 PROCSPEED:GOTO230
 460 PROCMODE:GOTO230
 470 PROCTRANS:GOTO230
```



```
  480 PROCSAVE:GOTO420
  490 K%=L%:L%=0:PROCDEL:W%=V%:V%=0
  500 H%=GET-48:IFH%<1ORH%>9L%=K%:GOTO
230
  510 PROCREC:V%=W%:L%=K%:GOTO230
  520 PROCLIST:GOTO150
  530 X%=C%
  540 Y%=N%(X%,1):FORZ%=1TO4:N%(X%,Z%)
=0:NEXT
  550 IFN%(X%+1,1)=Y%+1X%=X%+1:GOTO540
  560 PROCDEL:D%=1:GOTO230
  570 D%=1:R%=0:PRINTTAB(1,2);"  ";:GO
TO230
  580 VDU23;11,255;0;0;0;CLS:*FX4,0
  590 IFERR<>17AND ERR<>0THEN REPORT:P
RINT ERL
  600 END
  610 DEFPROCNOTE(Y%)
  620 PRINTTAB(6,P%(Y%,T%));CHR$134;N$
(Y%,T%);CHR$135;TAB(13,P%(Y%,T%));"O";
  630 ENDPROC
  640 DEFPROCREC
  650 IFR%=0D%=1
  660 Q%=0:PRINTTAB(0,1);CHR$129;C%;CH
R$135;"  ";
  670 PROCCHAN:N%(C%,2)=P%(L%,4):N%(C%
,3)=H%:N%(C%,4)=L%
  680 X%=N%(C%,1):N%(C%,1)=1:PROCPLAY(
C%):N%(C%,1)=X%
  690 R%=0:PROCDEL:PROCNOTE(L%)
  700 IFD%=3D%=1:Q%=1
  710 C%=C%+1:IFC%=SZ%-2 THEN PRINTTAB
(0,0);CHR$136;CHR$129;"FULL";:C%=SZ%-3
  720 ENDPROC
  730 DEFPROCSOUND
  740 *FX15,0
  750 SOUND1,V%,P%(L%,4)+A%,6
  760 IFD%>1SOUND2,V%,N%(C%-1,2)+A%,6
  770 IFD%=3SOUND3,V%,N%(C%-2,2)+A%,6
  780 IFD%>1ENDPROC
  790 IFN%(C%,1)>100SOUND2,V%,N%(C%+1,
2)+A%,6
  800 IFN%(C%,1)>500SOUND3,V%,N%(C%+2,
2)+A%,6
  810 ENDPROC
  820 DEFPROCPLAY(U%)
  830 Z%=V%:IFN%(U%,2)=0 Z%=0
  840 SOUNDN%(U%,1),Z%,N%(U%,2)+A%,N%(
U%,3)*E%
  850 ENDPROC
  860 DEFPROCCHORD
  870 IFI%<>H%D%=D%+1
  880 IFQ%=1PRINTTAB(25,1);"Chord full
!";:FORX=1TO1500:NEXT:PRINTTAB(25,1);S
PC11;:R%=0:ENDPROC
  890 R%=1:PRINTTAB(1,2);"Ch";:PROCDEL
:PROCNOTE(L%):ENDPROC
  900 DEFPROCVOL
  910 PRINTTAB(20,1);"Enter Volume (1-
9)";
  920 V%=GET-48:IFV%<1ORV%>9THEN920
  930 V%=-(V%+6):PRINTTAB(20,1);SPC18;
  940 ENDPROC
  950 DEFPROCSPEED
  960 PRINTTAB(20,1);"Enter Speed (1-9
)";
  970 E%=GET-48:IFE%<1ORV%>9THEN970ELS
EE%=10-E%
  980 PRINTTAB(20,1);SPC18;
  990 ENDPROC
 1000 DEFPROCMODE
 1010 PRINTTAB(20,1);"Enter Mode :";TA
B(21,2);"1 = Sharps";TAB(21,3);"2 = Fl
ats";TAB(21,4);"3 = Neither";
 1020 S%=GET-48:IFS%<1ORS%>3THEN1020
 1030 T%=S%:IFT%=3T%=1
 1040 FORX%=1TO4:PRINTTAB(20,X%);SPC18
;:NEXT
 1050 ENDPROC
 1060 DEFPROCDEL
 1070 FORY%=0TO23:PRINTTAB(6,Y%);CHR$1
35;"  ";TAB(13,Y%);"  ";:NEXT
 1080 IFR%=0ENDPROC
 1090 Y%=C%:REPEATY%=Y%-1:IFY%<0Y%=0
 1100 X%=N%(Y%,4):PROCNOTE(X%):UNTIL(N
%(Y%,1)MOD&100<2)OR(Y%=0)
 1110 ENDPROC
 1120 DEFPROCCHAN
 1130 IFD%=1N%(C%,1)=1:ENDPROC
 1140 IFD%=2N%(C%,1)=&102:N%(C%-1,1)=&
101:ENDPROC
 1150 IFD%=3N%(C%,1)=&203:N%(C%-1,1)=&
202:N%(C%-2,1)=&201:ENDPROC
 1160 DEFPROCSAVE
 1170 VDU31,18,2:S=OPENOUT("TUNE")
 1180 PRINT#S,C%,S%,V%,E%
 1190 FORX=0TOC%:FORY=1TO4:PRINT#S,N%(
X,Y):NEXTY,X
 1200 CLOSE#S:PRINTTAB(18,2);SPC21;
 1210 ENDPROC
 1220 DEFPROCLOAD
 1230 PRINTTAB(22,1);"Loading";:S=OPEN
IN("TUNE")
 1240 INPUT#S,C%,S%,V%,E%
 1250 FORX=0TOC%:FORY=1TO4:INPUT#S,N%(
X,Y):NEXTY,X
 1260 CLOSE#S
 1270 T%=S%:IFT%=3T%=1
 1280 ENDPROC
 1290 DEFPROCLIST
 1300 VDU14:CLS
 1310 PRINTTAB(20,1);"SPOOL (Y/N)? ";:
S%=GET:PRINT
 1320 IFS%<>ASC("Y")AND S%<>ASC("y")TH
ENS%=0ELSES%=1:*SPOOL SPTUNE
 1330 FORX%=0TOC%:PRINT;X%;" SOUND ";N
%(X%,1);",";
 1340 IF N%(X%,2)=0 THEN PRINT "0"; EL
SE PRINT ;V%;
 1350 PRINT",";N%(X%,2)+A%;",";N%(X%,3
)*E%:NEXT
```

```
 1360 IFS%=1THEN *SPOOL
 1370 PRINT""Press any key to continue
":*FX15,0
 1380 X=GET:VDU15:CLS
 1390 ENDPROC
 1400 DEFPROCTRANS
 1410 IFH%=25A%=A%+4
 1420 IFH%=26A%=A%-4
 1430 IFH%=27A%=0
 1440 IFA%=0PRINTTAB(0,3);"      ";
 1450 IFA%<>0PRINTTAB(0,3);"Tr";A%/4;"
";
 1460 ENDPROC
 1470 DEFPROCWRITE(X,Y,W$)
 1480 LOCALM:FORM=1TOLEN(W$)
 1490 PRINTTAB(X,Y+M-1);CHR$133;MID$(W
$,M,1);CHR$135;
 1500 NEXTM:ENDPROC
 1510 DATA0,1,1,2,2,3,3,4,5,5,6,6,7,8,
8,9,9,10,10,11,12,12,13,13,14,15,15,16
,16,17,17,18,19,19,20,20,21,22,22,23,2
3
 1520 DATA0,0,1,1,2,2,3,4,4,5,5,6,7,7,
8,8,9,9,10,11,11,12,12,13,14,14,15,15,
16,16,17,18,18,19,19,20,21,21,22,22,23
 1530 DATA0,1,0,1,0,1,0,0,1,0,1,0,0,1,
0,1,0,1,0,0,1,0,1,0,0,1,0,1,0,1,0,0,1,
0,1,0,0,1,0,1,0
 1540 DATA193,189,185,181,177,173,169,
165,161,157,153,149,145,141,137,133,12
9,125,121,117,113,109,105,101,97,93,89
,85,81,77,73,69,65,61,57,53,49,45,41,3
7,33
 1550 DATAB,A#,A,G#,G,F#,F,E,D#,D,C#,C
,B,A#,A,G#,G,F#,F,E,D#,D,C#,C,B,A#,A,G
#,G,F#,F,E,D#,D,C#,C,B,A#,A,G#,G
 1560 DATAB,Bb,A,Ab,G,Gb,F,E,Eb,D,Db,C
,B,Bb,A,Ab,G,Gb,F,E,Eb,D,Db,C,B,Bb,A,A
b,G,Gb,F,E,Eb,D,Db,C,B,Bb,A,Ab,G
```

POSTBAG

BRAIN TEASER

Dear Sir,

In Gareth Suggett's article on poles of numbers (Brain Teaser, BEEBUG vol.1 no.10 p.8) he says that 6174 is reached which ever 4-digit number is initially chosen. But there are nine numbers which do not give 6174; they are the numbers in which all four digits are the same (e.g. 1111,2222,etc). My daddy agrees with me. I did not use a computer to find this out.

Juliet Hounsome (aged 10).

EPSON MX80 DUMP

Dear Beebug,

You may already know but I thought you might be interested to learn that the MX80 Screen Dump - perhaps not surprisingly - also works on an Epson FX80.

Tony Briselden.

BLOWN FUSES IN TAPE RECORDERS

"A member of the Cossor Computer Club experienced a problem with his tape recorders which turned out to be caused by the connecting leads blowing the internal fuse. R.L. Carson of the Cossor Computer Club explains:

"I understand that the BBC (Acorn) computer is no longer delivered with a cassette-recorder connecting lead. This results in the need for the user to obtain (or make-up) a suitable lead.

One of our members bought a lead, which was advertised as being suitable for the BBC computer. When he plugged it into the recommended Ferguson cassette recorder, the cassette recorder ceased to work. He returned the recorder to the supplier, who exchanged it. When he connected the replacement recorder it also ceased to work.

Upon examination, it became clear that the supply fuse in the cassette recorder was blowing. The problem was caused by the fact that the connecting lead had jack plugs with metal covers: the remote control jack socket of the cassette recorder happens to carry the 6 volt supply (for the cassette recorder motor) and the adjacent socket carries 0v. The jack sockets on the recorder are so close that the covers of the jack plugs can touch one another as they are plugged in. The result can be that an excessive current flows from the 6 volt supply and - bang - the fuse blows. The problem was solved by the replacement of the remote jack plug (at the recorder end of the lead) by one with a plastic cover. So, beware of connecting leads with metal covers on the jack plugs, which might at first sight appear to be suitable!".

NON LINEAR PRINTING ON THE EPSON MX80

Mr A Dennis writes, "I am using a model B (0.1 OS) with an EPSON FT3 printer. I have bought a screen graphics dump programme, (modes 0 & 4) which takes approx 1 min. to dump to the printer, but it only uses approx. half of the 80 column width and the proportions are altered. I am writing Civil Engineering programmes which require a graphics dump routine to use the full paper width and if possible to maintain the screen proportions. Can you tell me a) if this is possible b) where I can buy a suitable programme. At present I am not conversant with machine code and quite frankly I do not have the time to investigate the possibilities on my own. I would be very grateful if you could help.

In the single density mode the EPSON prints 60 dots to the inch horizontally and 72 dots to the inch vertically. Therefore an allocation of one pixel per dot is bound to lead to distorted printing. To get an undistorted print would require 5 dots per pixel horizontally and 6 dots per pixel vertically; this is a print 27 ins wide and 21 ins high!

It would of course be possible to omit every 5th dot horizontally or to double print every 6th row of dots, but these would introduce other distortions.

This size of screen dump has a 20% distortion. It is possible, by using the double density facility on the EPSON to obtain a print that is only 7% distorted and is 8 ins wide by 7 ins high. The method is to use 3 double density dots per pixel (Modes 1 and 4, or 6 dots per pixel in Modes 2 and 5. In mode 0 this size is not possible without losing the 640 pixel per line definition) horizontally and 2 dots per pixel vertically. Using the same technique the size of our linear print could be brought down to a mere 13.5 ins wide by 10.5 ins high. The only answer for those requiring linear prints is to obtain an EPSON MX82, which prints 72 dots per inch horizontally and vertically. This should give two sizes of linear print, 4.5 ins by 3.5 ins and, on its side, 9 ins by 7 ins. [Powys Lybbe]

WHAT CASSETTE BUG?

"I enclose a copy of a letter I have recently received from Acorn concerning my BBC Micro. I would particularly draw your attention to the paragraph concerning the 'cassette bug' or lack of it.

It has taken 13 months since delivery of the machine to get the power supply changed and I fully expect it to take a further 13 months to prise a full spec ULA out of them. I have given up with my local dealer in Bridge-North having motored over 160 miles in repeated return trips to them in attempts to get the machine sorted out.

I can tell you that dealing with Acorn is better than any Adventure Game and it runs in Real Time, and I do mean REAL TIME.

I can claim to have introduced four of my colleagues to BEEBUG membership; we are all agreed that the magazine is excellent value for money. Keep up the good work" R.Glyn Jones.

Paragraph from Acorn letter: "CONTRARY TO YOUR CLAIMS THAT THE NEW O.S WILL CURE THE 'CASSETTE BUG' OUR TECHNICAL DEPARTMENT INFORM ME THAT THERE NEVER WAS A CASSETTE BUG, ONLY THAT SOME USERS WERE EXPERIENCING SLIGHT PROBLEMS IN CONNECTING THEIR CASSETTES CORRECTLY AND TO THIS END I ALSO ENCLOSE AN INFORMATION SHEET COVERING THE MORE COMMON CASSETTE PROBLEMS".
J.R. Fuller (Customer Support Adviser - ACORN Computers), Jan '83.

[Ed. Acorn's reply should of course have been dated 1st April. If any proof was needed that there really is a cassette bug in OS 0.1, we quote a note from Richard Russell of the BBC, whose name appears on the specification for the BBC micro. It was Richard who wrote the cassette bugs fix program for us back in July (BEEBUG vol.1 no.4 p.21). Acorn have since published this fix in Acorn User.

Quote: "THE CASSETTE FILING SYSTEM IN THE BBC MICROCOMPUTER (OPERATING SYSTEM VERSION 0.10) HAS A COUPLE OF RATHER UNFORTUNATE 'BUGS' WHICH AT BEST CAUSE ANNOYANCE AND AT WORST PREVENT THE USE OF CASSETTE DATA FILES IN CERTAIN CIRCUMSTANCES".
Richard Russell BEEBUG July '82]

ACORNSOFT GAMES REVIEW

by Alan Webster

Alan Webster reviews four extremely good games from Acornsoft. Thanks are due to Thomas and John Webb for their assistance with the Arcadians review.

TITLE: ARCADIANS PRICE £9.95

For me this is the best of the four Acornsoft programs reviewed in that it has been particularly well thought out. When loaded the program displays instructions, high scores, points awarded, gives demonstrations of the game and repeats the process until you choose the type of game that you require (very much like a dedicated arcade game).

The game involves shooting down hoardes of aliens(!) that swoop down in groups and bomb your missile blaster. The game gets very difficult after a few sheets of aliens have been killed. It is a truly addictive game, and represents good value for money.

TITLE: SNOOKER PRICE £9.95

Representing a game of snooker on a micro such as the Beeb is definitely not easy, especially with 8 coloured balls and a green table. Acornsoft have managed to reproduce the colours extremely well, although the green ball is cyan and the pink and brown can get mixed up if you're not careful. It plays to almost perfect rules and is very easy to control. It rates well above any other snooker or billiards program I've seen and will keep you happy for a long time. One point I must stress is that this is a game that requires a COLOUR display.

TITLE: INVADERS PRICE £9.95

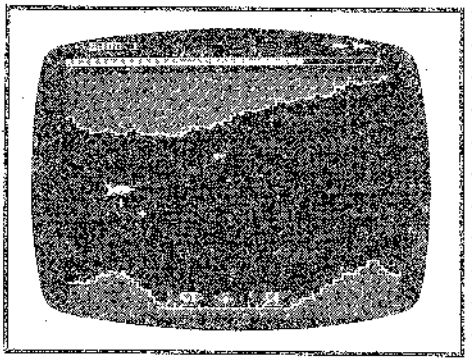
This is a good variation of the popular arcade game which runs in mode 1 and presents the player with various types of creatures such as an octopus(?), a crab, and other less recognisables.

The program runs smoothly and has 3 skill levels, the third of which has bombs which follow your base. The game is very imaginative and as with all Acornsoft games it is fun to play. (Does anybody know what comes after the rows of crocodiles?).

TITLE: ROCKET RAID PRICE £9.95

This is Acornsoft's version of 'SCRAMBLE' and as with most of their games it is a faithful copy of the original.

You pilot a spacecraft which travels through rocky caverns avoiding enemy rockets, fizzers etc. At the same time you must fire at fuel dumps to replenish your ever-dwindling supply.



The game requires great dexterity which means that it will take a long time before you fully master it.

Some points against this game are that the 'bomb' key is next to the escape key and this causes much frustration when the game suddenly ends; the landscape never changes from game to game; the missiles always attack in the same place and the screen shudders when a rocket takes off.

Overall however, it must be said that ROCKET RAID is an extremely good game, with some of the best graphics that you are likely to find.

SECRETS OF PROGRAM SAVING

by David Tall

In this article David Tall gives some very useful ideas on program saving, and on how to make back-up copies. His discussions are generally directed to cassette based programs, though most are applicable to disc.

It is a good plan to make back-up copies of all programs; you never know when an accidental erasure will leave you lamenting after it is too late. For instance, a common fault is to put tapes on top of the television where the powerful magnetic force forming the TV picture can ruin them. Even the static electricity in an atmosphere lacking humidity can wipe them out.

The simplest way of making copies of a program on cassette is tape to tape. The five second signal at the beginning of each program conveniently lets you set the volume level. However, if you have a disc drive or you only possess one tape recorder, the only method is to save the program from computer memory. As indicated by Andrew Donald (BEEBUG vol.1 no.9 p.7), should the program be anything other than straight Basic the usual SAVE command will not suffice. If there is machine code or data after a Basic program then you will only save the Basic, not the rest.

The universal solution for saving programs is provided by the *OPT1,2 command, briefly indicated on page 398 of the User Guide. Type *OPT1,2 and then, whether the program is in Basic, machine code or a mixture of the two, *LOAD it. When the loading is finished you will obtain a message like the following:

```
PROGRAM  0A  0A12  00000E00  00000FBA
         pages bytes  load      start
```

The first two blocks of numbers you will recognise as the number of pages and the number of bytes that normally feature in the loading message. The two additional pieces of information are the load and start address. They are eight digit hexadecimal numbers. Usually the first four numbers are 0000 but sometimes they are FFFF. It does not matter which; they are ready for future expansion. For our purposes we only require the last four. To save the program simply type:

```
*SAVE "PROGRAM" 0E00+0A12 0FBA
```

Note the change in order: load+bytes start. The plus sign is essential, so don't miss it out (see page 392 of the User Guide). If you wish you can omit the quotation marks and the leading zeros in any number so that a shortened version could be:

```
*S.PROGRAM E00+A12 FBA
```

You may also leave out the start address in two cases:

(i) if it is the same as the load address

(ii) if it is 801F.

In the first case start and load are taken equal by default, in the second the program is in Basic, &801F being the location in ROM where Basic is initialised.

When you have *SAVEd the program as indicated, if you *RUN it then it will be located exactly where it was saved and run as machine code. It is a good idea to take a note of the numbers used in saving, particularly the start address, because the program may be CALLed using this location. For instance, CALL &FBA would run the given example. It is also a good idea to find the contents of locations PAGE and PAGE+1 and note them down. (Type in PRINT ?PAGE and PRINT ?(PAGE+1)). Should your program be in machine code and you need to BREAK it to stop it (as with many commercial programs), then you can often reconstitute it and start again, first by typing OLD and ignoring the Bad Program message (which simply says it isn't Basic) and then by putting back the original values in locations PAGE and PAGE+1. (Type ?PAGE= 'the original value in

PAGE'and similarly the ?(PAGE+1).) Unhappily some manufacturers use *KEY10 to put garbage into part of the program when the BREAK key is pressed so this method will not always work.

If the program is CHAINed then, regardless of the position in memory when it was saved, the program will be loaded starting at the current PAGE number and will be RUN in the normal way. Provided that it begins as a Basic program this will work. The instructions on some manufacturer's programs to *LOAD and then to RUN are therefore not necessary, they only need to be CHAINed.

Many programs have a dual facility by including as a single line of Basic the CALL to the location where the machine code starts. For instance the given example might have begun with

```
10 CALL &FBA
```

Such a program may be *RUN or CHAINed to run it. For this reason it is worth having a peep at the program to see if it can be used in this way.

Only on one machine code program to date, have these methods failed. This was because the machine code was copied from a chip starting from address zero. If *LOADed it begins to overwrite the parts of zero page which do the LOADing and gets hung up. In such a case, issue the command *OPT1,2 as usual and then *CAT the program. The details will be given as before.

The program is not loaded over zero page in normal usage. It must therefore be relocatable and in practise be *LOADed at another address by an auxilliary program. The method of saving a back-up copy is to save it at any convenient address. If, for sake of argument, its details were:

```
SNEAKY 03 0400 00000000 00000000
```

then you must make a note of its length (400) and the *LOAD it at a convenient address, say:

```
*LOAD SNEAKY 2000
```

The messages you get on loading will still relate to the original locations but the code will actually be loaded beginning at &2000. You now save it with the instruction:

```
*SAVE SNEAKY 2000+400
```

By using techniques such as these, hopefully you can make back-up copies of all your programs to guard against accidental erasure at some later date. You can get quite a number of programs on a long hi-fi cassette. But do keep them in a safe place away from your main copies. You don't want to lose them through the same catastrophe that might afflict your working versions!

We should add that some commercial programs have been saved without an obvious start address so as to prevent the use of *OPT 1,2. Also it is possible to protect disc-based software at various levels, making the copying process a very difficult one.

BRAIN TEASER - THE FLAGSTONE PROBLEM

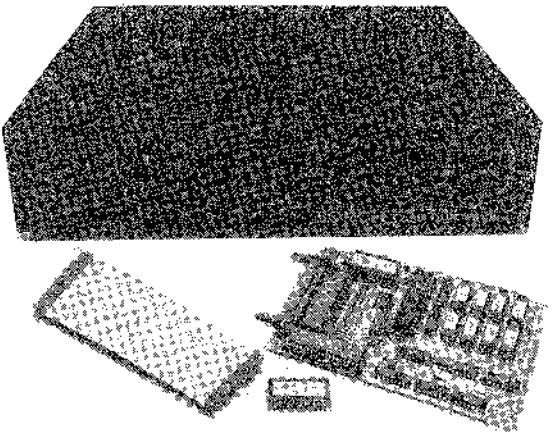
By Gareth Seggett

This is the second of our new series and a very interesting problem it is. Given 3 colours of paving slab, lay them in a path so that no pattern of colours is immediately repeated. For example no two consecutive slabs have the same colour; no two consecutive pairs have the same colours in the same order, and so on. Thus 1 2 1 3 1 2 1 is a length 7 solution, but cannot be extended because adding 1 repeats 1; adding 2 repeats 12; adding 3 repeats 1213. What is the longest path you can find?

TORCH Z80 DISC PACK REVIEW

by David Graham and Colin Opie

The Torch Disc Pack offers the owner of a BBC model B with disc interface, an upgrade consisting of dual 400k disc drives together with a second processor and an extra 64k of memory for only £780+VAT. This is not much more expensive than other dual drive units for the Beeb. As such, this package must look attractive to anyone whose pocket will stretch that far; but there are hidden snags, as we report below.



INSTALLATION

The unit is neatly packed, and comes with a utilities disc and ROM-based operating system. The manuals supplied were in fact manuals for the Torch computer rather than the Beeb-plus-add-on; and this gives a clue to what is about to take place. Your model B is about to be transformed into "almost a Torch computer"; perhaps not too surprising, since the Torch computer is based on a Beeb main board. Transformation involves, amongst other things, removing the power supply from the Beeb, and this needs considerable care. When you have made the conversion, and switch on your machine, it announces itself as a Torch computer, and if you have left the BBC disc filing system chip in your machine alongside its Torch counterpart, you now have a machine which can operate in two possible modes.

BBC WITH DISCS

In this mode the machine behaves just like a BBC model B with discs. The second processor is not used, and the extra 64k of memory cannot be employed for normal tasks such as running a BBC Basic program. There are two further snags to this mode of operation:

(1) you are not provided with a BBC utilities disc or disc manual for BBC operation - though BEEBUG vol. 1 no. 10 will help here in providing a disc formatter program and a summary of DFS commands.

(2) Because track buffering is handled differently on the Torch from that on the Acorn DFS, the disc heads click on and off excessively during reading and writing (especially with text files). It is not the noise itself that is worrying, but the wear to the mechanism.

ALMOST-A-TORCH

In this mode the drives no longer click excessively, and your machine behaves like a Z80-based Torch computer, with the Beeb handling input-output. In this state it will not read BBC discs, or run BBC Basic or assembler. Instead it has two immediate possibilities: Z80 machine code or something called CPN. This is a Torch version of CP/M, which is a universal I/O and file-handling system. Torch however do not offer any CPN support software - there is no currently available high level language of any description - and this is perhaps the biggest disincentive to buying the Torch Disc Pack. The CPN operating system does enable access to the OSBYTE, OSCLI, OSWRCH and OSWORD calls, but this is a little specialised for the average user.

CONCLUSION

As a piece of equipment in its own right, the Torch Z80 Disc Pack is well designed and packaged. Without more suitable software support its twin processing capability cannot be fully realised, and it is too expensive to use simply as a disc unit. Suitable CP/M languages are available but these are an added expense, and you will still have to write a number of procedures to customise any such commercially available package, if you want access to the Sound and Graphics of the Beeb. It will be interesting to see how Acorn's Z80 second processor add-on matches up to Torch's. Acorn are, we understand, developing a version of BBC Basic which will run on their Z80 unit.

Note - we understand that the Torch Disc Pack is now supplied with a manual of its own.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

FASTER CIRCLES AND REGULAR POLYGONS

Chris Eilbeck knows of a way to speed up the drawing of circles and polygons in Basic. Most programs which draw regular n-sided polygons or circles (large n) do so by repeated calculation of trigometric functions, which is slow and inefficient. For example, to draw a circle with centre at the point (500,500) with radius 300, the following could be used:

```
 90 MODE 0: TIME=0
100 X0=500: Y0=500
110 N%=100: T=2*PI/N%
120 R =300
130 MOVE X0+R,Y0
140 FOR THETA=0 TO 2*PI STEP T
150 DRAW X0+R*COS(THETA),Y0+
      R*SIN(THETA)
160 NEXT THETA
170 PRINT TIME
```

This takes 5.29 seconds. A much more efficient way is to use the well-known trigometric identities:

$$\cos(m+1)t = \cos(mt)\cos(t) - \sin(mt)\sin(t)$$
$$\sin(m+1)t = \sin(mt)\cos(t) + \cos(mt)\sin(t)$$

for m = 1,...n, which means only one sine and cosine need to be calculated before the loop is entered. The resulting program looks like this:

```
290 MODE 0: TIME=0
300 X0=500: Y0=500
310 N%=100: T=2*PI/N%
320 S=SIN(T): C=COS(T)
330 R=300: X1=R: Y1=0
340 MOVE X0+X1,Y0+Y1
350 FOR M%=1 TO N%
360 X=X1*C-Y1*S
370 Y=X1*S+Y1*C
380 DRAW X0+X,Y0+Y
390 X1=X: Y1=Y
400 NEXT M%
410 PRINT TIME
```

This takes only 1.83 seconds - almost three times faster!

NOTE If you have a 16K machine then you will need to change lines 90 and 290 respectively in the above two programs to read MODE4:TIME=0

SPACE INVADER PROMPT

The VDU23 call can be used very amusingly to re-define the prompt symbol '>' (ASCII 62) to be a 'space-invader'. Try the following:

```
VDU 23,62,60,126,219,255,126,60,36,66
```

Now you will see a space-invader as a prompt in all modes except Mode7. If you want to turn the whole character set into space invaders, try:

```
FOR A =32 TO 255:VDU 23,A,60,126,219,255,126,60,36,66:NEXT A
```

Again this does not have any effect in Mode7.

Alan Webster.

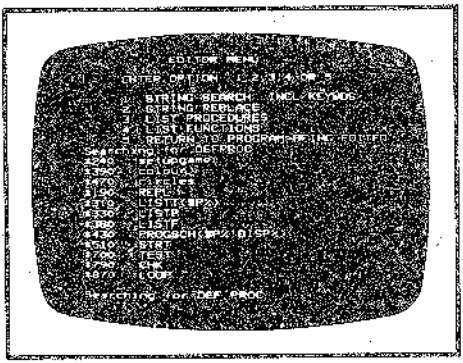
Program tested on O.S. 0.1 and 1.2

UTILITY EDITOR (16k/32k)

Program by Paul Otto. T. Burgess, A. Calcraft
Text by Adrian Calcraft

When developing a program much time can be wasted scanning the listing to locate a specific procedure, function or character string that needs to be changed. The program listed with this article is a mini editor, which can be loaded into the computer along with your development program, to allow you to:

1) List all occurrences of a specific string, including search for Basic Keywords.
2) Change all occurrences of a specific string to any new string that is equal or shorter in length than the original string, thus allowing you globally to change variable names.
3) List the starting line number and name of each procedure used.
4) List the starting line number and name of each function used.



BASIC KEYWORDS

As you have probably heard, when the computer stores a program in its memory, it converts all Basic Keywords (such as If, PRINT, SOUND, GOTO ...etc) into tokens. What this means is that rather than store all these, sometimes long, and frequently repeated words actually as they are spelt, it simply stores each of them as a 1 byte code (token).

This leads to much more economical use of storage space. However, it does result in a problem for string search programs; quite simply, searching for say, "Print" is a waste of time, as it has been stored in memory as 241. (see User Guide page 484)

A special feature of this editor is that option 1 will allow you to search for any string, INCLUDING a keyword, if you wish. This is achieved by special procedures which use the table at &806D (&8071 in BASIC II) in ROM to perform a conversion of the keyword in the string that you specify.

LIMITATIONS

The editor is stored for convenience just below Mode 7 HIMEM. Consequently running a development program which uses a Mode other than 7, with the editor also in memory, will overwrite the editor. This is no real problem as it is not necessary to run the user program while the editor is in situ, and in any case, the editor is easily reloaded. It should be noted that the string replace (option 2), will NOT locate or replace keywords.

Even with these minor restrictions the editor is a very useful tool, especially if you do not have the use of a printer with which to produce full program listings for inspection.

PROGRAM DETAILS

The program is in fact a combination of several procedures, sent in mainly by two members. We are most grateful to T Burgess for the first (PROCREPL) and to P Otto for (PROCLISTT PROCLISTP and PROCLISTF).

The procedures have been combined and are accessed via a menu, to create a single more powerful editor. However, if you decide that you are interested say, in only one of the procedures, they are easily separated.

BEEBUG INDEX FOR VOLUME 1 April 1982–March 1983

The symbol (S) after the page number indicates that this item is to be found in the 'Supplement'.

GENERAL CONTENTS

TITLE	ISSUE	PAGE
10 Questions	1	13
Book Survey	10	8(S)
Brain Teaser	10	8
Disc Roundup 1	9	19
Disc Roundup 2	10	25
FX Calls	1	16
FX Calls in Assembler	2	14
FX Calls - Undocumented	10	15
HAM List	10	3(S)
Function Key Labels	10	34
Letters	2	21
Local User Group Index	10	16(S)
Postbag	5	17
Postbag	9	28
Quick Quiz	9	7
Sound	1	10
User Guide Bugs	5	4
User Port on A & B (Part 1)	3	8
User Port (Part 2)	4	17
(Points Arising	5	25)
Wordwise Ideas 1	9	24
Wordwise Ideas 2	10	41

PROCEDURE/FUNCTION LIBRARY

TITLE	ISSUE	PAGE
Date Validation	10	39
Days Between Dates	5	28
(Points Arising	6	27)
Double Height Chars	6	18
Ellipse Drawing	1	18
Filled Circles	8	18
INPUT Function	4	24
(Points Arising	6	27)
Sort procedures	7	17
Yes/No Function	5	28

BEEBUG MAG

ARTICLES

TITLE	ISSUE	PAGE
3-D Rotation	10	3
Accented Letters	10	24
Analogue Upgrade	8	27
Assembler Guide	2	11
Assembler Guide (Part 2)	3	20
BBC Basics - Memory Maps	7	14
BBC Basics - Procedures	9	21
BBC Basics - Reaction Timer	6	3
BBC Basics - Various	8	14
BBC Basics - Windows	10	21
Debugging (Part 1)	5	10
Debugging (Part 2)	6	19
Debugging (Part 3)	7	19
Files (Part 1)	9	12
Files (Part 2)	10	19
Games Writing (Part 1)	2	17
Games Writing (Part 2)	3	17
Graphics	2	6
Graphics (Part 2)	3	11
Ideas on Animation	9	3
Ideas on Colouring	5	7
Key Define	4	5
(Points Arising	5	25)
Logic (Part 1)	5	21
Logic (Part 2)	6	15
Merging Programs	2	16
Micro Electronics Project	5	30
Multi Coloured Chrs	5	11
Negative Inkey	6	13
Notes on Program Saving	9	7
Paged ROMs on Upgraded A	10	18
Passing Filenames To MOS	10	31
Program Compacter	8	11
(Points Arising	9	31)
Program Transfer	4	27
Retrieving a Bad Program	8	3
(Points Arising	9	31)
Serial Port & RGB upgrade	7	25
Sound & Envelope (Part 1)	7	5
Sound & Envelope (Part 2)	8	22
Structuring (Part 1)	3	5
Structuring (Part 2)	4	11
Teletext	4	7
Tube	6	22
Unexplained Errors (See Debugging)		
Upgrading to Model B	1	19
User Port (Part 1)	3	8
User Port (Part 2)	4	17
(Points Arising	5	25)
User key Update	5	26
Using Files (See Files)		
Verifying Programs	5	5
Video Chip 6845	8	29
What To Do With 1.2	10	15

REVIEWS

TITLE	ISSUE	PAGE
Book Review - Cryer	2	3
Book Reviews - Hartnell	8	25
Johnson-Davis		
Book Reviews - Birnbaum	9	15
Cownie		
Prigmore		
Ruston		
Brief Review Of BBC Micro	1	5
Disc Review	8	6
(See Disc Roundup	9	20)
Joystick Review	7	10
More Joysticks	9	24
Printer Review	10	9
Software Reviews	4	14
Software Reviews	6	24
Software Reviews	8	20
Video & TV Review	3	13
(Points Arising	4	30)
Video & TV Review Update	7	30

HINTS AND TIPS

TITLE	ISSUE	PAGE
16k Test	5	10
(See also FX254	9	11)
3-D Letters	6	21
A/D Converter Upgrade	7	33
Abbreviated NEXT	8	16
Addresses	2	5
Amazing Sounds	4	22
(Points Arising	7	34)
Basic ROM bug	10	17
Baud Fix for 110 baud	3	10
Black on White (Hardware)	6	6
Black on White (Software)	5	30
Block Move	6	6
Break, OLD & NEW	8	10
Bugs in OS 0.1	3	7
Calculated GOTO Renumber	7	4
Caps Lock/Shift Lock	10	8
Cassette File Speeds	1	15
Cassette Problems	4	29
Cassette Problems	9	26
Cassette Voltage	8	34
Clear The Screen	1	15
Coloured Listings	9	31
Coloured Text	1	16
Coloured Text & Listings	6	30
Coloured Titles on Disc	10	39
Composite Video (Colour Fix)	2	5
Control Characters	10	32
Cursor Delete	3	26
Cursor Off	2	14

BEEBUG MAG

Debugging With EVAL	10	30
Defining Keys	9	20
Disable Escape & Break	3	7
Disc Roundup	9	19
Disc Space Economies	10	39
Disc-Cassette Break	9	31
ELSE Warning	7	27
FX calls in 0.1 OS	1	16
FX calls in 1.2 OS	10	15
FX18	9	20
FX254	9	11
(See also 16k test	5	10)
Free Memory	4	19
Function Key Recall	5	27
GET from RS423 on OS 0.1	4	30
HIMEM, LOMEM, TOP & PAGE	2	15
Hex Notes	8	16
Hex To Decimal	1	15
I/O Addresses	2	5
INKEY Timer	8	10
INPUTLINE	4	4
Integer Variables	5	9
Jumping out of FOR/NEXT	7	9
Last Line	9	20
Loading Times	8	24
Loud Audio	8	28
MOTOR	3	7
Machine Code at &E00	8	9
Make-shift Dollar Sign	10	34
Merging VDU statements	8	9
Mode 6 Background	3	7
More Lives In Rocket Raid	10	34
(Points Arising, see Vol 2 No 1)		
Moving Graphics Origin	9	31
Multi Program Store	8	10
Musical Data Statement	10	32
Musical Keyboard	9	26
(Points Arising	10	40)
Noisy Audio	8	31
Non-Accessible Filenames	10	30
Not True?	10	30
ON ERROR Bug	9	24
ON ERROR CRASH	7	16
OS Notes	5	6
OSBYTE &87	8	13
OSBYTE INKEY	4	3
Out of Sequence AUTO	5	27
Page Change for Discs	7	27
Peek & Poke Equivalents	2	14
Planetoid Restart	10	14
Plotting Bug in 0.1	10	17
Printer Control	4	13
Printer Interfacing	5	19
Printers (110 baud Warning)	2	15
Printers (110 baud Fix)	3	10
Procedure Anomaly	3	26
Program Length	10	6
Programs via 'phone	6	7
REMs in DATA	5	27
ROMs & EPROMs	2	15
Randomise	4	26
Read a Character	3	23
Renumber Query	8	19
Screen Addresses	2	5
(Blunders)	3	16)
(Points Arising	4	30)
Screen Scrolling	1	15
Self Validating GET	7	8
Sideways Scroll	9	14
Silent Games	10	8
Silly Renumber	4	22
Sony Profeel TV/Monitor	10	38
Sort Function	4	10
Sound Equivalence	7	29
Speed	4	6
Step Through Listing	4	4
String Handling	4	28
System Calls	9	7
THEN omission	3	7
Trapping Errors	5	24
User Defined Chrs	7	22
VDU Commands	1	15
Verify	3	25
Video Output - Colour Fix	2	5
What Mode?	8	41
(Points Arising	9	31)
Windows	1	15

UTILITY PROGRAMS

TITLE	ISSUE	PAGE
3-D Rotation (32k)	10	3
Bad Program Retrieval (16k)	8	3
(Points Arising	9	31)
Bugs Fix (16k)	4	21
(Points Arising	7	34)
Calendar Generator (16k)	6	29
Cassette Block 0 Bug (16k)	7	3
Compacter Revisited (16k)	9	25
Disc Formatter	10	29
Epson Screen Dump		
(See Screen Dump)		
Harmonograph (16k)	7	12
Masterfile (Software Library)	9	24
Memory Display (16k)	6	8
(Points Arising	7	34)
(More Points Arising	8	38)
Mini Text Editor V.2	7	31
(Points Arising	8	38)
Mini Wordprocessor	3	21
New Chr Set in Modes 2&5	7	9
Polygon (16k)	3	4
Program Compacter (16k)	8	11
(Points Arising	9	31)
(Compacter Revisited	9	25)
Screen Dump - Seikosha	4	23
Screen Dump - Seikosha (M/C)	9	9

BEEBUG MAG

Screen Dump - Epson	4	23	Single Key Memory Display	9	27
Screen Dump - Epson (M/C)	9	9	String Search (16k)	5	15
(Points Arising	10	40)	String Sort (16k)	6	9
Screen to Cassette	6	7	Transparent Loader (16k)	7	23
Seikosha Screen Dump			Wordwise Ideas	9	24
(See Screen Dump)					

VISUAL AND SOUND PROGRAMS

TITLE	ISSUE	PAGE
3-D Rotation (32k)	10	3
3-D Surface (16k)	1	9
Artist (32k)	8	39
Micro Sketch (16k)	10	40
Multi-Coloured Beeb (16k)	5	20
Patchwork (16k)	4	4
Rotating Cube (32k)	9	6
Screen Play (32k)	3	10
Screen Play (16k)	5	16
Ship 1 (32k)	9	5
Ship 2 (32k)	9	6
Spiroplot (16k)	9	11
Square Dance (16/32k)	10	7
Union Jack (16/32k)	6	21

GAMES PROGRAMS

TITLE	ISSUE	PAGE
3-D Noughts & Crosses (32k)	1	23
Aliens (16/32k)	6	31
Artillery Duel (16/32k)	10	33
Beebmaze (32k)	9	32
Bomber (16k)	2	23
Breakout (16/32k)	8	32
(Points Arising	9	31)
Careers (32k)	2	24
Chunky Invaders (16k)	4	31
(Points Arising	5	25)
(More Points Arising	6	27)
Five Dice (16k)	9	29
Hangman (16k)	5	33
Higher/Lower (16k)	5	31
(Points Arising	6	27)
Life (32k)	10	35
Maze Trap (32k)	3	24
Moon Lander (16k)	1	22
Moon Lander - Ideas	9	24
Racer (16k)	7	28
Space City (16k)	8	36
Starfire Update	6	27
Starfire Joysticks	7	34

BEEBUG MAG

TO SET UP THE EDITOR

Type in the editor as listed. To enable two programs to be in memory at the same time (the editor and the program you will use it on), the editor is stored at the top of available memory, out of the way. Obviously the actual location of this will differ between 16k and 32k machines. Therefore, if you have a 16k machine, alter line 10 to:

```
10 *KEY0PAGE=&3000|MRUN|M
```

Note that the default value of PAGE on the BBC machine is &1900 when a disc interface is fitted to the computer, (rather than &E00 on cassette-based machines).
If you have a disc system then change line 20 to:

```
20 page=&1900
```

Once you have typed in the program check your version against the original listing, given at the end of this article, and then save a copy to tape or disc, under the file name "EDITOR".

TO USE THE EDITOR

1) If you have a 16k machine type PAGE=&3000 <return>
   If you have a 32k machine type PAGE=&7000 <return>
2) Load and run the program using CHAIN"EDITOR". Running it will immediately set the red function key (f0) ready for later use.
3) Now press ESCAPE. This will stop the editor from running further.
4) Press function key 1. You are now ready to load in your development program in the normal way.
5) To invoke the editor use function key 0, which will take you to the editor menu.
6) Having finished using the editor select option 5 to continue with your program.

ABOUT THE EDITOR

When you run the editor you will be presented with a menu offering the 5 options. To choose an option simply use the relevant key, as indicated. All other entries are locked out. If you enter 1 or 2 you will be prompted for a search string. Simply enter your string and then press return. Option 2 will also prompt you for a replace string which you key-in in the same way.

As already mentioned, option 1 will allow you to search for strings including a keyword. To tell the computer that the relevant part of the string is a keyword, it must be enclosed in 2 slashes (/). eg:

```
To search for PRINT "HELLO"
 enter /PRINT/ "HELLO"
To search for REPEAT
 enter /REPEAT/
To search for A%=A%+1 :NEXT B%
 enter A%=A%+1 :/NEXT/ B%
```

As you see you must enter the search string exactly as it exists in the program, but adding / symbols around the keyword, if there is one.
ONLY 1 KEYWORD MAY BE SEARCHED FOR IN ANY 1 STRING

If the program can not find the keyword specified it will print an error message, and continue the search with the string exactly as you entered it.

If you choose options 3 or 4 no prompts are required and the search for procedures or functions is initiated immediately.

Having displayed the relevant line numbers (if there are any to display) the editor will prompt you to press return, upon which the menu will be redisplayed, ready for your next command. To return to your development program select option 5.
NOTE All options put the display in "paged mode". This prevents the located line numbers from being displayed, only to scroll off the top of the screen. If a full screen is displayed, pressting "shift" will display the next screenful.

IF YOU HAVE PROBLEMS

If upon running the editor you get an error message indicating that the computer has run out of room (eg. DIM space), it probably means that your version of the program has become longer than the original. This may be due to accidentally adding extra spaces. If this is so you can very easily overcome this by reloading the program slightly lower in memory (eg. &6F00 for 32k and &2F00 for 16k machines). If you have to do this, don't forget to alter line 10 to reflect the new start address.

ERROR MESSAGES

"Error try again". This is only issued by option 2 and indicates either that you have attempted to replace a string by a longer string, which is not allowed, or that the editor cannot find the program to edit.

"Invalid Keyword". This is displayed by option 1 and indicates that you have attempted to search for a Basic Keyword which has not been found in the table. eg /PRNT/ . If this is specified as a search argument in option 1, the error message will be given. The search will continue, however, for the actual string /PRNT/. ie no conversion will take place.

ABOUT THE KEYWORD CONVERSION

The keyword table replacement routine will only be able to locate the keywords exactly as they are stored in the ROM memory. You will find it useful to know that the token for INSTR assumes the first bracket, and so should be searched for as /INSTR(/. This also applies for LEFT$, which is searched for as /LEFT$(/ , MID$ which is /MID$(/ , POINT which is /POINT(/ , RIGHT$ which is /RIGHT$(/ , STRING$ which is /STRING$(/ and TAB which is /TAB(/ .

The following keywords are stored twice in the table, only the first entry is accessed by the conversion program. These are..PAGE, PTR, TIME, LOMEM, HIMEM.

If a search is requested for a string containing a keyword and nothing else, after conversion, the program will be searching for a single byte (the token). It is possible that this byte may occur in a different context within the program, eg as part of a line number. If this is the case, the program will erroneously report a line number. This happens in practice very rarely, and only on searches for strings containing just a keyword and nothing else.

Searching for GOTO followed by a line number, eg /GOTO/200 ,will not work. This is because the line number after a GOTO, is stored by the computer in coded form too.

A TIP FOR USERS WITH DISC SYSTEMS

You may find it more convenient to load and run this program from another program. This will save you having to set PAGE every time you decide to load the editor again. To this end save the main program as "EDITOR" and save the following two line program as "EDSTART".

```
10 PAGE=&7000
20 CHAIN "EDITOR"
```

So now to load and run the editor you need only to enter, CHAIN "EDSTART"

****NOTE**** Do not forget to customise the program to your machine if you have a 16k or disc system, as described above.
Finally, note that the variable 'page' occurs at several points in the program, and that it MUST be typed in lower case.

```
 10 *KEY0PAGE=&7000|MRUN|M
 20 page=&E00:REM Type page in lower case
 30 DIM T%12,R%12,P%250
 40 MODE7
 50 CLS:PRINT TAB(11,5)"EDITOR MENU"
 60 PRINT TAB(5,7)"ENTER OPTION..1,2,3,4
OR 5"
 70 PRINT TAB(5,9)"1..STRING SEARCH..INCL/
KEYWDS/"
 80 PRINT TAB(5,10)"2..STRING REPLACE"
 90 PRINT TAB(5,11)"3..LIST PROCEDURES"
100 PRINT TAB(5,12)"4..LIST FUNCTIONS"
110 PRINT TAB(5,13)"5..RETURN TO PROGRAM
BEING EDITED"
120 ON INSTR("12345",GET$) GOTO 130,140,
150,160,180ELSE 50
130 INPUT"SEARCH STRING.."$P%: PROCSTRT:
PROCLISTT($P%):GOTO 170
140 PROCREPL: GOTO 170
150  PROCLISTP: GOTO 170
160   PROCLISTF
170 INPUT"COMPLETED..PRESS RETURN" $P%:
GOTO50
180 PAGE=page:END
190 DEFPROCREPL
200 B%=page+1:Q%=B%:INPUT"SEARCH STRING.."
$T%:INPUT"REPLACE STRING.."$R% :PRINT"THE
FOLLOWING LINES ALTERED..": VDU14
210 IF ?(Q%)=&FF OR LEN($R%)>LEN($T%)
PRINT"Error, try again":VDU15:ENDPROC
220 REPEAT:V%=?(B%+2) :FOR X%=1 TO V%:?(Q%+
X%-1)=?(B%+X%-1):NEXT
230 REPEAT:IF ?(Q%+2)-4<LEN($T%) LOC=0:
GOTO270
240 $P%="":FOR C%=Q%+3 TO Q%+?(Q%+2)-2:$P%
=$P%+CHR$(?C%):NEXT:LOC=INSTR($P%,$T%):IF LO
C=0 GOTO 270 ELSE IF $T%=$R% PRINT 256*?(Q%)
+?(Q%+1):LOC=0:GOTO270
250 FOR C%=1 TO LEN($R%):?(Q%+LOC+1+C%)=
ASC(MID$($R%,C%,1)):NEXT:FOR C%=1 TO ?(Q%+2)
-LOC-LEN($T%)-2:?(Q%+LOC+LEN($R%)+C%+1)=?(Q%
+LOC+LEN($T%)+C%+1):NEXT
260 ?(Q%+2)=?(Q%+2)+LEN($R%)-LEN($T%):PRIN
T 256*?(Q%)+?(Q%+1)
```

```
  270 UNTIL LOC=0:Q%=Q%+?(Q%+2):B%=B%+V%
  280 UNTIL (?(B%-1)=&0D AND ?(B%)=&FF) OR
B%=LOMEM:?(Q%)=&FF
  290 VDU15:ENDPROC
  300 REM
  310 DEFPROCLISTT($P%):PROCPROGSCH($P%,0):P
RINT:ENDPROC
  320 REM
  330 DEFPROCLISTP
  340 PRINT"Searching for DEFPROC":PROCPROGS
CH(CHR$&DD+CHR$&F2,-1)
  350 PRINT"Searching for DEF PROC":PROCPROG
SCH(CHR$&DD+" "+CHR$&F2,-1)
  360  ENDPROC
  370 REM
  380 DEFPROCLISTF
  390 PRINT"Searching for DEFFN":PROCPROGSCH
(CHR$&DD+CHR$&A4,-1)
  400 PRINT"Searching for DEF FN":PROCPROGSC
H(CHR$&DD+" "+CHR$&A4,-1)
  410 ENDPROC
  420 REM
  430 DEFPROCPROGSCH($P%,DISP%)
  440 K%=LEN($P%):VDU14:A%=@%:@%=8:AST$="*"
  450 IF DISP%@%=6:AST$="*"
  460 I%=page-1
  470 REPEAT:I%=I%+1:IF?I%=P%?0 PROCLOOP
  480 UNTIL ?I%=&0D AND I%?1=&FF:VDU15:PRINT
:@%=A%
  490 ENDPROC
  500 REM
  510 DEFPROCSTRT
  520 A%=P%-1:Q%=P%+LEN($P%):Y%=0
  530 FORN%=1 TO2
  540 X%=Y%
  550 REPEAT:A%=A%+1:UNTIL ?A%=&2F OR A%>=Q%
  560 IF ?A%=&2F Y%=A%
  570 NEXT
  580 IF X%=0 OR X%=Y% ENDPROC
  590 X%=X%+1:B%=Y%-X%
  600 PROCTEST
  610 IF F%=&FF PRINT"INVALID KEYWORD":ENDPR
OC
  620 A%=P%:P%=P%-1
  630 REPEAT P%=P%+1:UNTIL ?P%=&2F
  640 ?P%=F%: Z%=P%+1
  650 REPEAT P%=P%+1:UNTIL ?P%=&2F
  660 REPEAT: P%=P%+1:?Z%=?P% :Z%=Z%+1: UNTI
L P%=Q%
  670 P%=A%:$P%=MID$($A%,1,Z%-A%)
  680 ENDPROC
  690 REM
  700 DEFPROCTEST
  710 F%=0
  720 D%=&806D
  730 REPEAT
  740 IF ?D%=?X% PROCCHK
  750 IF F%=0 REPEAT:D%=D%+1:UNTIL ?D%>&7F:
D%=D%+2
  760 IF D%>&8348 OR ?D%>?X% F%=&FF
  770 UNTIL F%<>0
  780 ENDPROC
  790 DEFPROCCHK
  800 C%=D% :F%=1
  810 FOR N%=0 TO B%-1
  820 IF X%?N% <> ?C% F%=0
  830 C%=C%+1:NEXT
  840 IF ?C% < &80 OR C%?1 > &24 AND C%?1 <>
 &43 F%=0
  850 IF F%=1 F%=?C%
  860 ENDPROC
  870 DEFPROCLOOP
  880 C%=1:FOR J%=I%+1 TO I%+K%: IF ?J%=P%?
C% C%=C%+1 ELSE J%=K%+I%
  890 NEXT:IF C%<K% ENDPROC
  900 FOR J%=I%-1 TO I%-254 STEP-1:IF ?(J%-2
)=13 PRINT AST$;?J%+?(J%-1)*256,;: J%=I%-254
:IF DISP%X%=I%+K%:PRINT" ";:REPEAT:Y%=?X%:PR
INT CHR$(Y%);:X%=X%+1:UNTIL Y%=13 OR Y%=58
OR Y%=61:PRINT
  910 NEXT:ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

REMOVING SPACES AT THE END OF LINES

When entering or editing lines of BBC Basic, unwanted spaces may be accidentally inserted at the end of any line. These do no harm other than increase the size of the program, but this is often undesirable. To remove these spaces it is necessary to highlight them in some way, and this can be easily done using any mode other than Mode 7.

Contrary to implications in the User Guide (see p.384), any character with an ASCII value greater than 31 can be redefined, without having to reserve memory or 'explode' the character store using special FX calls, (the only restriction being that not more than 32 different definitions can be obtained without these other operations being performed), and this is true for any operating system. The important point is that there is only space for 32 definitions and that ASCII codes 32-63, 64-95 etc. up to 224-255 map on to the same definition storage area.

By redefining the space character (ASCII 32) to be a large white rectangle, a listing will show them up clearly and the COPY facility can be used to copy a line up to where the extra spaces start, hence 'deleting' the unwanted spaces.

To see this effect type in a few lines of Basic that have extra spaces on the end. Now type in immediate mode the following:

```
MODE6:VDU 19,0,1,0,0,0:VDU 23,32,255,255,255,255,255,255,255,255
```

List your program again and you will be able to see all those spaces!

Alan Webster.

NEWCOMERS START HERE
SOME EXPERIMENTS IN MODE 7 - THE TELETEXT MODE

by David Graham

We have received requests from some members for articles for absolute newcomers to computing. Here is an offering from David Graham in which he assumes very little previous knowledge. The article covers the use of PRINT and TAB statements, and some control codes in mode 7.

The BBC micro has 8 possible display modes, they differ in the size of printed characters, the number of colours available, and the so-called graphics 'resolution' - this is just the degree of fineness or chunkiness of the graphics picture. The modes which give the most colours and the highest resolution are generally those which use up the most memory. Mode 7, which has a very low resolution picture capability but offers 7 colours, uses up only about 1000 units (or bytes) of memory, whereas mode 1 takes up over 20,000 (and leaves precious little memory for any programs to reside in).

Because it uses so little memory in a machine where memory space is at a premium, Mode 7 is a particularly important one for BBC users. It is also called the Teletext mode. This is because it uses all the same conventions as Teletext systems such as Prestel or Oracle; and it is by using mode 7 that it is possible to dial into Prestel with a small add-on to your Beeb.

Mode 7 is different from graphics modes such as 1, 2, 4 and 5 in an important respect; you cannot use commands like PLOT and DRAW to draw lines or fill triangles. Instead everything must be put on to the screen using the ordinary PRINT statement or equivalent. Do not however think that Mode 7 is an uninteresting one; quite the reverse.

GETTING INTO MODE 7

When you turn on your machine (or hit the BREAK key at any time) the machine automatically defaults to mode 7. If you are in another mode, you can change to mode 7 by typing MODE 7 <return> (where <return> means press the key marked RETURN). There is an abbreviation for this command. It is MO. 7 <return>. Most keywords in BBC Basic have abbreviations. A full list is given on pages 483 and 484 of the User Guide (or on the BEEBUG Reference Card).

PRINT STATEMENTS

The most commonly used way of printing text on the screen is the PRINT statement. You place in quotes after it the text to be printed - eg type:

```
PRINT "This is a test" <return>
```

The computer should do as requested, and print the text below the command line.

TAB

There is a TAB statement in BBC Basic, rather like a typewriter tab. It allows you to place your printed text anywhere on a line or anywhere on the whole screen in fact. Try typing:

```
PRINT TAB(10)"HELLO" <return>
```

You should see that the text will be indented by 10 characters. If you wish to specify which line the text is to appear on, you can do this as follows:

```
PRINT TAB(15,4)"TEST"
```

This places the word TEST fifteen characters along on the fourth line down the screen. If there happens to be something on the screen in that position, it will be overwritten.

40 x 25 SCREEN

In each mode, there is a fixed number of character positions available for printing. In Mode 7 there are 25 lines on which text may be placed (labelled 0 to 24), and each line can accommodate up to 40

characters (the positions being numbered from 0 to 39). If you try to print further along than character 39, the text will automatically spill over on to the next line. If the screen is full, it will scroll upwards to leave space below.

COLOURS

There are 7 possible colours in Mode 7 if you include white, and each has associated with it a code number. These are listed below.

Code	Colour		
129	Red	132	Blue
130	Green	133	Magenta
131	Yellow	134	Cyan
		135	White

To get a line of coloured text, you must insert an invisible colour code character before the text. One way to do this is with the CHR$ command. Try the following:

PRINT CHR$(130)"GREEN TEXT" <return>

The text will be printed in green (providing you are in Mode 7). The code telling the computer to put the text in green is printed invisibly in the character position immediately to the left of the word GREEN, as the computer has printed it. You can test this by using the ↑ (up arrow) and COPY cursors to copy the line of green text. Unless you start copying at the space before the word GREEN, the text will copy over as white. You can experiment with the other colour codes; and they may be used in combination. Try:

PRINT CHR$(129)"RED TEXT";CHR$(133)"MAGENTA TEXT" <return>

COLOURED BACKGROUND

It is possible to alter the background colour on any line from natural black to one of the 7 colours. To do this you use code 157. This means "change background colour to the current text colour". Thus:

PRINT CHR$(129);CHR$(157);CHR$(132)"BLUE TEXT ON RED BACKGROUND"<return>

will give a background band of red with the text in blue. This is because the code preceding the "set background code" is 129 - the code for red - and this is followed by the code that sets the foreground. Some quite striking effects can be achieved in this way.

VDU COMMANDS

It is possible to abbreviate the sequence of CHR$ commands on the BBC micro using a single "VDU" command. This is used outside a print statement. The above example can now be replaced with the shortened version:

VDU129,157,132:PRINT"BLUE TEXT ON RED BACKGROUND" <return>

OTHER EFFECTS

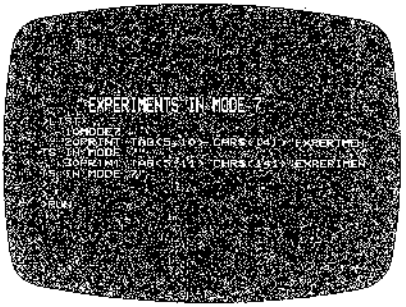
A number of other effects are possible in Mode 7, all achieved by the interplay of control codes. For example there is one for flashing characters (code 136 = Flash On, 137 = Flash off), and another for double height characters. With the latter you must print the same line of text twice, as you can see if you type:

PRINT CHR$(141) "DOUBLE HEIGHT" <return>

You need to enter it twice, as follows

PRINT CHR$(141)"DOUBLE HEIGHT": PRINT CHR$(141)"DOUBLE HEIGHT" <return>

With a little ingenuity, you should be able to produce flashing double height text on a coloured background.



USE WITHIN A PROGRAM

The ideas suggested above may easily be incorporated into programs of your own design. If you have not written a program before, it is done by preceding statements with a number - usually increasing in steps of 10. For example enter the following:

10 MODE 7 <return>.

20 PRINT TAB(10,10) CHR$(129);CHR$(157); CHR$(132) "BLUE TEXT ON a red background" <return>

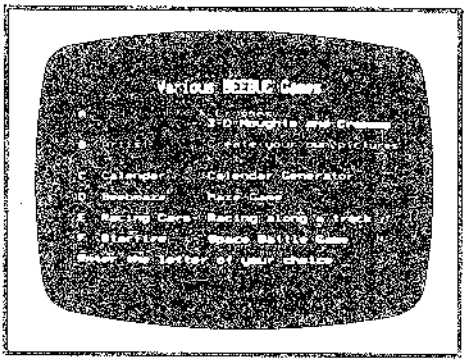
Each time you type RUN <return>, the sequence will be executed in line number order. The command LIST <return> (or L.<return>) will list the program. It may be saved on a cassette with SAVE"COLOURS" <return>, and loaded back in with LOAD "COLOURS" <return>, or just LO."" <return>.

For more ideas on mode 7 see BEEBUG vol.1 no.4 p.7 and the User Guide p.150.

Program tested on O.S. 1.2

DISC MENU PROGRAM

by John Yale



The February issue of BEEBUG explained how to run a program from disc automatically when SHIFT and BREAK are pressed simultaneously. One use of this is to present a menu of the programs available on the disc, which may then be selected with a single key rather than having to type CHAIN"filename".

The program listed below allows a menu of up to 19 filenames together with program descriptions to be presented after <shift><break>. The user then presses a single key to chain in the required program file. As well as giving single key selection, the program allows ample description of each program, and helps rescue the user from the very limited seven character filenames of the Beeb's DFS.

The program has been written to allow very simple setting up for new menus. Just follow these steps:

1) Type the program in or LOAD it from disc.
2) Edit line 1750 to be the title of the Menu eg. Games, Music, Graphics.
3) Edit lines 1800 to 2250 with as many programs as reqired (up to 19). Each DATA statement MUST contain three items for each program: the filename, the name to be displayed and the description.
4) SAVE"!MENU"
5) *BUILD"!BOOT"
   and enter the text:
   CHAIN"!MENU" <return>
   then press escape.
6) *OPT 4,3

The disc is now ready for use. Hold down <shift> whilst pressing and releasing <break>, and the menu will be displayed. Select a program by typing one of the letters shown.

This program can only load Basic programs, so to use it with machine code a small Basic header should be used. A single line containing *RUN filename is all that is required.

```
  100 REM Menu program
  110 REM 22 Jan 1982   J.Yale
  115 ON ERROR GOTO 5020
  200 READ TITLE$
  300 DIM F$(20),N$(20),D$(20)
  400 I=0
  500 REPEAT
  600 I=I+1
  700 READ F$(I),N$(I),D$(I)
  800 UNTIL F$(I)="DONE"
  850 MAX=I-1
  900 MODE 7
 1000 PROCCENTRE(TITLE$)
 1100 FOR J=1 TO MAX
 1150 IF MAX<10 THEN PRINT
 1160 letter$ = CHR$(64+J) + "."
 1170 colour$ = CHR$(129+(J-1)MOD 6)
 1200 PRINT letter$;colour$;N$(J);TAB(14);
"- ";D$(J)
 1300 NEXT
 1350 VDU 28,0,24,39,VPOS
 1400 PRINT""Enter the letter of your choi
ce ";
 1405 REPEAT
 1410 A=GET
 1415 PRINT CHR$(A)
 1420 REQ=A-64
 1430 IF REQ>0 AND REQ<=MAX THEN CHAIN F$(
A-64)
 1500 PRINT""Select a letter from A to ";C
HR$(MAX+64);" ";
 1505 UNTIL FALSE
 1510 END
 1520
```

```
1600 DEF PROCCENTRE(T$)
1600 DEF PROCCENTRE(T$)
1605 PRINT'
1610 L%=LEN(T$)/2
1620 FOR Y=1 TO 2
1630 PRINT CHR$141;CHR$134;TAB(20-L%);T$
1640 NEXT
1650 ENDPROC
1700
1740 REM Enter Title of Disc here
1750 DATA Various BEEBUG Games
1760
1790 REM Filename,Game-title,Description
1800 DATA 3DOXO,3-D Noughts & Crosses,3-D
Noughts and Crosses
1900 DATA ARTIST,Artist,Create your own p
ictures
2000 DATA CALEND,Calendar,Calendar Genera
tor
2100 DATA MAZE,Beebmaze,Maze Game
2200 DATA RACER,Racing Cars,Racing along
a track
2220 DATA STFIRE,Starfire,Space Battle Ga
me
2230
2240
2250
2260 REM Add extra games above here
2270 REM This next line must be present
2300 DATA DONE,DONE,DONE
5000
5010 REM Error routine
5020 REPORT
5025 IF ERR<200 PRINT ERL :END
5030 PRINT"  ( ";F$(A-64);" )"
5035 X=INKEY(500)
5040 RUN
5050 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TV SHAKE FIX

Neil Morrison has a useful tip for anyone whose TV display shakes. He has found that use of VDU23,0,3,X,0;0;0; where X is between 18 and 25 may stop the shaking (it works on his anyway!). On his TV the top few lines of the display shake violently in modes 0-3 until this command is typed in.
[This form of the VDU23 command programs the 'horizontal sync' register of the video controller chip and is therefore improving the 'locking' of the TV electronics to the computer's signal. Different values to those above may therefore be necessary for your particular TV or monitor. (See BEEBUG vol.1 no.8 p.29). Ed]

PROGRAMMABLE FUNCTION CALLS

When a Function or a Procedure in Basic is called its name must follow the FN or PROC keyword. This is all very well, but if some kind of selection has to take place, at one or more points in a program, as to which routine to access the programming can become a little tedious and long-winded.

David McTernan knows of a method to call Functions 'dynamically' by using a variable for the name part of the call, the appropriate Function then being accessed through a dummy EVAL expression. This provides a convenient and concise mechanism for accessing various subroutines.

By way of example suppose we use 'A$' as the variable for the name of the Function (without 'FN'), then the following program will call the Function 'demo':

```
10 A$ ="demo"
20 ARG$="FN"+A$
30 DUMMY=EVAL(ARG$)
.
.
100 END
110 DEF FNdemo
120 P."Dynamic call"
130 = 0
```

'A$' could also contain an argument list as long as the function is defined as having those arguments. Note that the above mechanism is really a form of Procedure call as the 'result' of the 'Function' is not really required.

Program tested on O.S. 0-1 and 1-2

DOUBLE HEIGHT CHARACTERS IN ALL MODES

by Bjorn Grand

PROCEDURE FUNCTION LIBRARY

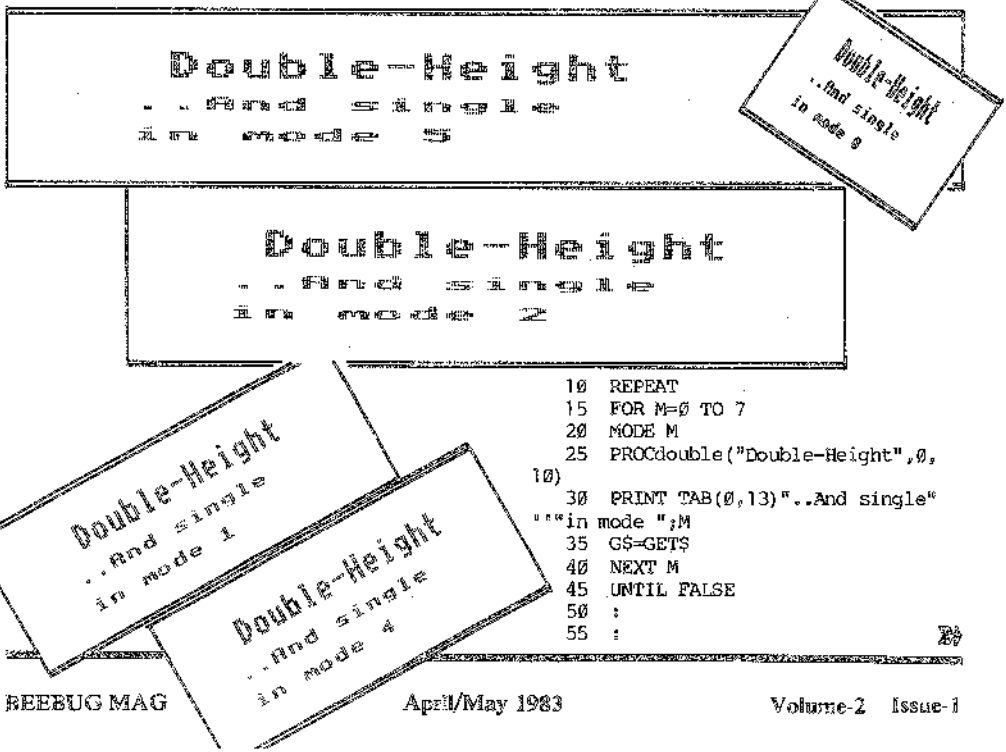
One of the big advantages of the Teletext mode, is the ability to use double-height characters. With this little procedure, which can be incorporated in any program, you will be able to obtain double-height characters in any mode. With it you can produce some nice bold lettering, especially in modes 2 and 5 where the character width is greatest. The syntax is quite simple:

PROCdouble("HELLO",15,10)

will print the word 'HELLO' at (15,10), in double-height, no matter which mode you are in!

The routine works by first checking to see which mode is currently active by using OSBYTE 135 (as described in BEEBUG vol.1 no.8 p.41 - 'What MODE?'). If the screen is found to be in Mode 7, the program branches to a different routine within the procedure and uses the "normal" method, with CHR$(141). If however the screen is in one of the other modes, then the OSWORD 10 call is used which returns the pattern of a character, and this pattern is then used to define two new characters, CHR$(240) for the upper half and CHR$(241) for the lower half. This is then repeated for each of the characters in the string to be printed.

The procedure itself is from lines 1000 on. The first part of the program is a demonstration which shows how the double-height characters look in the different modes. You will notice that in the Text Only modes (Modes 3 and 6) there is a black line separating the two halves of the double-height characters, due to the spacing between lines in these modes. To see this a little more clearly you could change the background colour in these modes by using: VDU 19,0,1,0,0,0




```
10  REPEAT
15  FOR M=0 TO 7
20  MODE M
25  PROCdouble("Double-Height",0,
10)
30  PRINT TAB(0,13)"..And single"
"in mode ";M
35  G$=GET$
40  NEXT M
45  UNTIL FALSE
50  :
55  :
```

```
1000  DEF PROCdouble(A$,K,L)
1010  LOCAL N
1020  A%=135
1030  IF(USR(&FFF4)AND &FF0000) DIV
&10000 = 7 THEN 1190
1040  :
1050  REM Modes 0-6
1060  A%=&A:X%=0:Y%=&A
1070  D=&A00
1080  FORN=1 TO LEN(A$)
1090  B$=MID$(A$,N,1)
1100  ?D=ASC(B$)
1110  CALL (&FFF1)
1120  VDU23,240,D?1,D?1,D?2,D?2,D?3
,D?3,D?4,D?4
1130  VDU23,241,D?5,D?5,D?6,D?6,D?7
,D?7,D?8,D?8
1140  PRINT TAB(K+N,L);CHR$(240);
TAB(K+N,L+1);CHR$(241)
1150  NEXT N
1160  ENDPROC
1170  :
1180  REM Mode 7
1190  PRINT TAB(K,L);
1200  FOR N=0 TO 1
1210  PRINT TAB(K);CHR$(141);A$
1220  NEXT
1230  ENDPROC
```

POINTS ARISING

SCREEN TO CASSETTE (v.1 no.6 OCT)

The simple screen to cassette routines published as above do not (and cannot easily) take account of so called "hardware scrolling" used by the Beeb. If you are using these routines, you either need to ensure that the screen has not been scrolled after it was last cleared, or to force it to do a so called 'software scroll'. The latter is achieved by defining a text window equal to the full size of the screen (see BEEBUG vol.1 no.10 p.21). This must be accomplished after a mode change or a clear screen command (CLS).

COMPACTED PROGRAMS

Can we please enforce a point, made in the first of the two articles on the 'Compacter' utilities, that compacted programs can NOT be later edited using the COPY key. In the same way a listing of a compacted program cannot be typed in again directly. The reason, as stated, is because important spaces between keywords already stored as tokens are no longer present.

We would also ask you NOT to send compacted programs to BEEBUG as part of an article; but rather the original sources. Once we list them in the magazine there may be some lines which look correct but will not run.

BEEBMAZE (v.1 no.9 FEB)

Some people have found an error in the program Beebmaze listed last month. The error occurs if the score ever falls below -99 and the length of the score (number of digits plus negative sign) is greater than 3. This causes the display to scroll upwards. The simple cure for this is to change the value of 11 in the second TAB statement at line 2180 to 10. [In our tests we never scored that low ! Ed].

EPSON MX DUMP (v.1 no.9 FEB)

Several readers have written in to us pointing out that if a graphics program redefines the graphics origin and does not reset it before the screen dump is used, only part of the picture is printed. The remedy is to execute VDU 29,0,0 before dumping.

MORE LIVES ROCKET RAID

There was a typographical error in our 'more lives in Rocket Raid' hint in the March issue. The restart address is &117B, and not &1178 as stated. We apologise for this - we have now had an opportunity to test John Harris's hint, and it certainly works.

Program tested on O.S. 0.1 and 1.2

IMPLEMENTING PSEUDO-OPERATIONS AND MACROS USING THE BBC ASSEMBLER

by David C. Nichols

The BBC Computer comes equipped with an assembler, although anybody who has used a 'real' assembler will have noticed that the assembler in the BBC Computer has no psuedo-operations (except OPT). Psuedo-operations are the name given to operations that are directed at the assembler not the machine.

For example the OPT statement tells the ASSEMBLER what to do about errors and what listing is required, it generates no machine code. Common psuedo-operations found in assemblers might be:-

BYTE - generate an 8 bit constant
WORD - generate a 16 bit constant
TEXT - generate a text string
RESV - allocate space for a buffer

The June issue of BEEBUG (vol.1 no.3 p.20) explained a method of implementing some of these operations, and the October issue of Personal Computer World (p.186) showed another. Both these methods suffer from the disadvantage of having to leave the assembler, do some Basic and then return to the assembler (this requires that an OPT statement be repeated every time you re-enter the assembler).

There is a third, and I think, much better method that does not require you to leave the assembler. The method exploits the fact that the operands for the assembler statements are Basic expressions. For example it is perfectly legal to write:

```
LDA #SIN(RAD(45))*100
```

as an assembler statement, (even more amusing is LDA #GET). [Nice one - Ed].

This feature would not be very useful in itself except that BASIC expressions can contain calls to user defined functions, and that the operand to the OPT statement can also be an expression (yielding a value between 0 and 3). Consider the following demonstration program:-

```
10 DIM space 100
20 FOR pass=0 TO 3 STEP 3
30 P%=space
40 [ opt pass
50   brk
60   opt FNbyte(255)
70   opt FNtext("This is a string")
80   opt FNbyte(0)
90 ]
100 NEXT pass
110 CALL space
120 END
130 DEF FNtext(s$)
140  $P%=s$
150  P%=P%+LEN(s$)
160 =pass
170 DEF FNbyte(b%)
180  ?P%=b%
190  P%=P%+1
200 =pass
```

The program illustrates the exploitation of the features discussed (lines 60 to 80). The function definitions in lines 130 to 200 implement two common psuedo-operations in a manner very similiar to that shown in the June BEEBUG; the functions finish by returning a number in the range 0 to 3 (=pass) which is required for the OPT statement (redundantly, but this doesn't matter).

The user functions invoked in this way can contain any legal Basic statements, including more assembler statements! This latter point allows us to write macros (a macro is a group of statements that can be "called" during the assembly process, this is quite different to a subroutine). The following program examplifies the implementation of a macro call:

```
10 DIM space 100
20 FOR pass=0 TO 3 STEP 3
30 P%=space
40 [ opt pass
50.loop
```

```
   60  opt FNmdex(3)
   70  beq ok
   80  bpl loop
  100  brk
  110  opt FNbyte(255)
  120  opt FNtext("Number not
divisible by 3")
  130  opt FNbyte(0)
  140.ok
  150  brk
  160  opt FNbyte(0)
  170  opt FNtext("Number
divisible by 3")
  180  opt FNbyte(0)
  190]
  200 NEXT pass
  210 REPEAT
  220  INPUT''"Input a number
  between 1 and 127 " X%
```

```
230 UNTIL X%>=1 AND X%<=127
240 CALL space
250 END
260 DEF FNtext(s$)
270  $P%=s$
280  P%=P%+LEN(s$)
290 =pass
300 DEF FNbyte(b%)
310  ?P%=b%
320  P%=P%+1
330 =pass
340 DEF FNmdex(c%)
350 LOCAL i%
360 FOR i%=1 TO c%
370  [opt pass
380   dex
390  ]
400 NEXT
410 =pass
```

Lines 260 to 330 are the same as lines 130-200 in the previous program. The function in lines 340 to 410 implements a macro that generates assembler statements that decrement the X register by the number supplied in the parameter to the macro (c%). The program above, when run, also illustrates a "quirk" of the assembler; the statement invoking the macro (line 60) is listed after the statements it generates, this can be confusing but unavoidable.

Notes:

1. The MOS expects an error number followed by a message terminated by a zero after a BRK instruction.
2. The assembler treats upper and lower case letters in the instruction mnemonic as the same, eg. LDA=lda=LDa etc.

With the above ideas and a bit of ingenuity it is possible to implement structured programming macros that enable you to write in assembler high level language control constructs (e.g. IF ---- THEN ---- ELSE ----).

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SCROLL COMMANDS

Those of us who 'come up from the gutter' - that is, started computing on a ZX-80 or 81 - know how useful the 'scroll' statement can be, both for games and for serious applications. Fortunately, it is quite easy to implement this on the Beeb:

A) SCROLL DOWN (all modes): VDU 30,11
This will scroll the screen down one line, leaving the cursor at (0,0). VDU 30 is 'cursor home'. VDU 11 is 'cursor up' but, since the cursor is already at the top line, this will instead force the screen one line down.

B) SCROLL UP (modes 3,6,7 - text modes): VDU 31,09,24,10
This will scroll the screen up one line, leaving the cursor at (0,24). It works in a similar way to the above.

C) SCROLL UP (modes 0,1,2,4,5 - graphics modes): VDU 31,0,31,10
Scrolls the screen one line up, leaving the cursor at (0,31). For more information about how these calls work, see User Guide pp.377-389.

Thanks to Bjorn Grand for this hint, which came all the way from Denmark.

Program tested on O.S. 0·1 and 1·2

COLOUR BAR CHART GENERATOR (32k)

by Tim Powys-Lybbe

Tim Powys-Lybbe presents an extremely useful and easy to use program for generating colour bar charts of any size or shape. If you ever require data represented in visual form, here is an excellent way to do it.

The program Bar-Chart is intended primarily for generating screen displays, though with a suitable screen dump, these could be printed out or saved to tape or disc(see BEEBUG no 9 for suitable screen dumps). The program allows you to design a multicoloured bar chart through a series of simple commands, and text may be written around or on the chart.

The program is written in a fairly condensed manner, though with line-number-plus-SPACE separators between procedures for legibility. It takes up just under 4k of memory which gives you about 5k to add any extras you want. The program will fit onto a Micro with disc interface without any modification, though of course there is then only about 2k of free memory.

For saving the display on to cassette or disc, or obtaining a printed copy, a procedure PROCScreen is called from line 440 of PROC Exit. A description of this procedure is given at the end of this article under the sub-heading 'Exit'.

Extensive use is made of the BBC's procedure and repeat-until facilities. An alphabetical directory of the procedures used is given to assist in following the program, not to mention any debugging, should you make transposition errors when typing the program in.

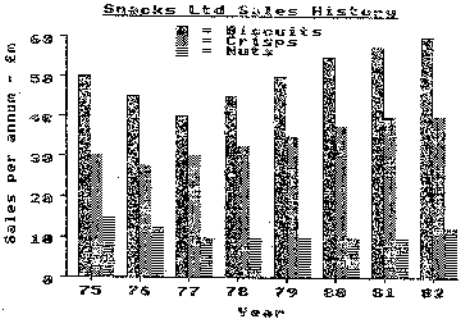
INSTRUCTIONS FOR USE

1. MARK OUT THE CHART AREA

(PROC Limits)

Check that the CAPS LOCK is on while making the Bar Chart and move the cursor with the cursor keys.

Enter T at the top of the area you choose, O at the bottom left or origin of the area and R at the right hand side of the area. The cursor keys normally move the cursor 8 pixels at a time. This can be changed to single pixel movements by pressing the TAB key. Pressing it again returns to 8 pixel movements. The cursor keys can be held down together to give diagonal movements. There is no check to ensure that you put T above O and R to the right of O; if you do not then the program will give odd but predictable results. The chart can be any size, anywhere on the screen. The axes are drawn automatically after entry of T, O, R.



2. DEFINE AND DRAW THE BARS, THEIR SHADINGS AND COLOURS

(PROC BarPositions)

This procedure asks a series of questions to which the answers are as follows:

(a) "How many different bars?" This refers to the types of bars; a maximum of four types are allowed. For each of the four you can choose any one of four patterns and any one of three colours.

(b) "What character for bar 1?" Enter either A, B, C or D to choose a predefined character used for bar shading. Characters B, C and D are patterned and can appear to give another colour in combination with the background colour.

(c) "What colour for bar 1?" Enter 1, 2 or 3 for red, yellow or white. These are the default Mode 1 colours

and can be changed using the VDU 19 command; flashing colours are permitted.

(d) "Chars to be half/full width? (0/1)" Enter 0 for half width or bars 4 pixels wide and 1 for full width or bars 8 pixels wide.

(e) "How many sets of these bars?" Enter any positive integer number. If the number is too large for all the sets of bars to fit on your chart area, you will be told so, and will have to start again with the number of bars. One purpose of the 'sets' facility is to permit charting different items on the same chart. For instance you may want to chart the sales of Model A, Model B, disc interface and Econet type BBC Micros over a six month period. This would need four bar types and six sets of these bar types for the six months. This could be done in full width bars on a screen-wide chart but if the chart were over twelve months, half width bars would have to be used for the 48 bars to fit on the screen. See instruction 3 below for more details.

(f) "How many pixels between bars?" Enter any integer number in the range 0 to 99. This decides the gap to the left of each bar.

(g) "How many pixels between sets?" Enter any integer number in the range 0 to 99. This decides the minimum gap to the right of each set of bars. The sets will in fact be spaced evenly on the X or horizontal axis so will usually be separated by more than this minimum gap.

(h) "Maximum value of Y?" (PROC Maxima) Enter any number in the range 0.00001 to 100000. This number is then used to accurately scale the entries for the individual bars.

(i) "Set no 1; Bar no 1. Bar value?" (PROC BarValues) Enter any number in the range from 0 up to the maximum value entered in the last item. The number can include decimals but cannot be longer than six figures including the decimal point. The bars are then drawn to scale automatically.

3. MAXIMUM NUMBERS OF BARS

The maximum chart width is 302 pixels. This permits the following maximum numbers of full and half width bars:

	Full	Half
Zero pixel separation	37	75
Single pixel separation	33	60

The number of bars is the product of the bar types and the sets of those types.

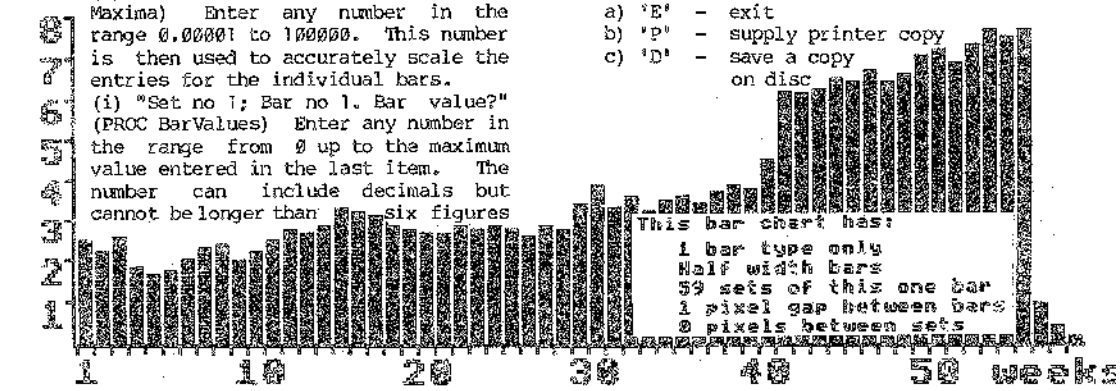
4. TEXT MODE (PROC Text)

At this stage you can write any character in upper or lower case anywhere on the screen; this includes the defined characters with the colours that you chose for the bars, which are written with the f1 up to f4 keys, depending on your number of bars. The cursor can be moved with the cursor and TAB keys as above (instruction 1). At the end of a line the cursor will automatically go to the next line down, separated by four pixels, as in the Micro's Mode 3. Auto-repeat works normally, as do the RETURN and DELETE keys, though the latter additionally deletes the current character.

5. EXIT

Once text has been entered, the exit procedure is called. Although no prompt or echo of input is given (so as not to affect the display), you can type any of the following:

a) 'E' - exit
b) 'P' - supply printer copy
c) 'D' - save a copy on disc



The program listing currently only supports the 'E' and 'D' options. If you want a 'P'rinter option then all you need do is add on a suitable screen dump routine for your particular printer. BEEBUG vol. 1 no. 9 contains suitable machine code screen dump routines for both the Epson and Seikosha printers. The copy sent to disc under the 'D' option will be given the name 'SCREEN' and the screen display can therefore be re-loaded with MODE1:*LOAD "SCREEN" at any desired time.
The same name is always used in the saving of the screen contents and therefore it is important on disc based systems to rename this file, or change the name in the program listing, if a second screen is to be saved on the same disc.

Note: The program Bar-Chart is a much reduced version of a single routine from an elaborate screen processor package which will be available through BEEBUGSOFT in the near future. The package allows diagrams of all kinds, including bar charts, pie charts and graphs, as well as cursor-drawn figures to be placed on the screen, and subsequently saved to printer, cassette or disc. Please do NOT write for further details, these will appear in the magazine as soon as they are available.

PROCEDURES		
Home	800	Puts the cursor at the top left of the screen.
Initialise	160	Initialises variables, sets up defined characters, function keys and background colour.
Input	920	Allows only characters specified in A$ to be input up to a specified length and displays them at a specified position on the screen.
InKeyCursor	880	Main procedure to flash and move the cursor and to return with keyboard input plus current cursor position.
Letter	480	Places text in lines format on the screen.
Limits	760	Inputs the markers for the axes.
Maxima	830	Inputs maximum bar value permissible and hence determines the scale of the Y axis.
Message	480	Sets up text window at the bottom of the screen for the command line and issues VDU 4 instruction.
Screen	990	For user-supplied screen dump routine.
Scrub	940	Deletes present cursor position in text mode, moves back to previous text cursor position and deletes that.
Text	850	Main procedure for writing text and bar legends on the screen.
Vecblo	340	Mnemap for X(as) and X(do).

```
  10 REM Bar Chart in Colour
  20
  30
  40 REM Copyright T F Powys-Lybbe 1983
  50
  60
  70 ONERRORGOTO130
  80 MODE1
  90 PROCInitialise
 100 PROCBarChart
 110 PROCText
 120 PROCExit
 130 VDU4:*FX4
 140 END
 150
 160 DEF PROCInitialise:DIMA%(3,1):DIMB%(3,
1):XLO%=36:YLO%=64:XHI%=1248:YHI%=1020:MOVE0
,36:DRAW1280,36:M=1:GCOL0,3:COLOUR3:PROCHome
:CS=8:J=0:*FX4,1
 170 E$="0123456789":F$=E$+".":A$=F$:C$=F$:
VDU23,255,255,255,255,255,255,255,255,255,23
,254,0,65,34,20,8,20,34,65,23,224,255,255,25
5,255,255,255,255,255:*KEY0|!|A
 180 VDU23,225,255,0,255,0,255,0,255,0,23,2
26,170,85,170,85,170,85,170,85,23,227,170,17
0,170,170,170,170,170,170,5,19,0,4,0,0;:*KEY
1|!|B
 190 *KEY2|!|C
 200 *KEY3|!|D
 210 *KEY4|!|E
 220 ENDPROC
 230
```

```
  240 DEF PROCCursor:GCOL4,3:IFGD=0 MOVEX,Y:
VDU255 ELSE MOVEX-16,Y+16:VDU254
  250 MOVEX,Y:IFM=1 M=0 ELSE M=1
  260 GCOL0,3:ENDPROC
  270
  280 DEF PROCInkeyCursor:REPEAT:K%=INKEY(0)
:J=J+1:IFJ>40 J=0:PROCCursor
  290 IFNOT(K%=-1) IFM=0 PROCCursor
  300 IFK%>135ANDK%<140 PROCCursorKeys
  310 IFK%=9 ANDCS=1 CS=8 ELSEIFK%=9 CS=1
  320 UNTILNOT(K%=-1):ENDPROC
  330
  340 DEF PROCYesNo:A$="YN":PROCInput(1,POS,
VPOS):ENDPROC
  350
  360 DEF PROCCursorKeys:REPEAT:IFINKEY(-26)
 IF NOT(X<=XLO%+(CS-1)*4) X=X-4*CS
  370 IFINKEY(-122) IF NOT(X>=XHI%-CS*4) X=X
+4*CS
  380 IFINKEY(-42) IF NOT(Y<=YLO%+4*CS) Y=Y-
4*CS
  390 IFINKEY(-58) IF NOT(Y>=YHI%+4*(CS-1))
Y=Y+4*CS
  400 MOVEX,Y:K%=INKEY(0):UNTILK%<136 OR K%>
139:ENDPROC
  410
  420 DEF PROCDelete:MOVEX,Y:GCOL0,0:VDU255:
GCOL0,3:MOVEX,Y:ENDPROC
  430
  440 DEF PROCExit:MOVE0,36:PLOT7,1280,36:PR
OCCommand(""):PROCScreen:VDU6,12,16,20,26:
*FX12
  450 *FX15
  460 ENDPROC
  470
  480 DEF PROCMessage:VDU4,28,0,31,39,31,12:
ENDPROC
  490
  500 DEF PROCEndMessage:VDU26,5:MOVEX,Y:END
PROC
  510
  520 DEF PROCInput(length,X,Y):LOCALF%:F%=0
:C$="":REPEAT:B$=GET$:IFINSTR(A$,B$)>0 C$=LE
FT$(C$+B$,length):F%=1:PRINTTAB(X,Y)C$;
  530 IFB$=CHR$(127) AND length>1 C$=LEFT$(C
$,LEN(C$)-1):PRINTTAB(X,Y)C$;" ";:VDU31,X+LE
N(C$),Y
  540 UNTILF%=1AND(B$=CHR$(13) ORlength=1):C
=VAL(C$):ENDPROC
  550
  560 DEF PROCCommand(A$):PROCMessage:PRINT
A$;:PROCEndMessage:ENDPROC
  570
  580 DEF PROCAxes:FORI=0TO2:X=A%(I,0):Y=A%(
I,1):PROCDelete:NEXT:MOVEXorigin,Ymax:DRAWXo
rigin,Yorigin:DRAWXmax,Yorigin:ENDPROC
  590
  600 DEF PROCHome:X=XLO%:Y=YHI%:MOVEX,Y:END
PROC
  610
  620 DEF PROCBarChart:PROCLimits:PROCAxes:P
ROCBarPositions:PROCMaxima:PROCBarValues:END
PROC
  630
  640 DEF PROCBarPositions:PROCMessage:REPEA
TCLS:PRINT"How many different bars? ";:A$=ES
:PROCInput(2,POS,0):BarNo%=C:IFBarNo%=0 ORBa
rNo%>4 VDU7:UNTILFALSE
  650 FORI=1TOBarNo%:REPEAT:CLS:A$="ABCD":PR
INT"What character for bar ";I;" ";:PROCInpu
t(1,POS,0):B%(I-1,0)=159+ASC(C$):PRINT" ";CH
R$(B%(I-1,0));" OK? Y/N ";:PROCYesNo:UNTILC$
="Y"
  660 REPEAT:CLS:A$="123":PRINT"What colour
for bar ";I;" ";:PROCInput(1,POS,0):B%(I-1,1
)=C:COLOURB%(I-1,1):PRINT" ";CHR$(B%(I-1,0))
;" OK? Y/N ";:COLOUR3:PROCYesNo:UNTILC$="Y":
NEXT
  670 CLS:PRINT"Chars to be half/full width?
 (0/1) ";:A$="01":PROCInput(1,POS,0):Width=C
:REPEAT:CLS:PRINT"How many sets of these bar
s? ";:A$=E$:PROCInput(2,POS,0):Sets%=C:UNTIL
Sets%>0
  680 D$="How many pixels between ":A$=E$:CL
S:PRINTD$+"bars? ";:PROCInput(2,POS,0):Bgap%
=VAL(C$)*4:CLS:PRINTD$+"sets? ";:PROCInput(2
,POS,0):Sgap%=VAL(C$)*4
  690 A%=(Xmax-Xorigin-4)/(BarNo%*((Width+1)
*16+Bgap%)+Sgap%):IFSets%>A% CLS:PRINT"Hit S
PACE. Bad sets/bars. Max sets=";A%;:VDU7:REP
EATUNTILGET=32:UNTILFALSE
  700 CLS:PRINT;Sets%;" sets of ";BarNo%;" b
ars. OK? Y/N ";:PROCYesNo:UNTILC$="Y":ENDPRO
C
  710
  720 DEF PROCBarValues:REPEAT:FORI=1TO Sets
%:FORJ=1TO BarNo%:PROCMessage:REPEAT:CLS:PRI
NT"Set no ";I;"; Bar no ";J;". Bar value? ";
:A$=F$:PROCInput(6,POS,0):A=C:IFA<0 ORA>MaxY
 VDU7:UNTILFALSE ELSEPROCEndMessage:UNTILTRU
E
  730 RatioY=(Ymax-Yorigin)/MaxY:X=Xorigin+(
Xmax-Xorigin)/Sets%*(I-1)+(J-1)*16*(Width+1)
+J*Bgap%+4:FORK=32TO32+A*RatioY STEP32:Y=Yor
igin+K:MOVEX,Y:GCOL0,B%(J-1,1):PRINTCHR$(B%(
J-1,0)):GCOL0,3:IFWidth=0 X=X+16:PROCDelete
:X=X-16
  740 NEXT:Y=Yorigin+A*RatioY+32:PROCDelete:
NEXT:NEXT:ENDPROC
  750
  760 DEF PROCLimits:FORI%=0TO3:B%(I%,0)=0:N
EXT:GD=1:REPEAT:PROCCommand("Bar Chart: Ente
r T, O, R in CAPS"):PROCInkeyCursor:A$=CHR$(
K%):IFA$="O" Xorigin=X:Yorigin=Y:B%(0,0)=1:V
DUK%:A%(0,0)=X:A%(0,1)=Y
  770 IFA$="R" Xmax=X:B%(1,0)=1:VDUK%:A%(1,0
)=X:A%(1,1)=Y
  780 IFA$="T" Ymax=Y:B%(2,0)=1:VDUK%:A%(2,0
)=X:A%(2,1)=Y
  790 A=1:FORI%=0TO2:IFB%(I%,0)=0:A=0
  800 NEXT:IFA=0 UNTILFALSE
  810 UNTILA=1:GD=0:ENDPROC
  820
  830 DEF PROCMaxima:CLS:A$=F$:PRINT"Maximum
 value of Y? ";:PROCInput(6,POS,0):MaxY=C:EN
DPROC
  840
  850 DEF PROCText:XLO%=0:XHI%=1280:PROCHome
:PROCCommand("Enter Text: f0 key will exit")
:REPEAT:PROCInkeyCursor:IFK%>31 ANDK%<127 PR
OCLetter
  860 IFK%=127 PROCScrub
  870 IFK%=13 ANDY>YLO%+40 Y=Y-40:X=XLO%
  880 IFK%>129 ANDK%<134 PROCBars
  890 MOVEX,Y:UNTILK%=129:ENDPROC
  900
```

```
  910 DEF PROCLetter:PRINTCHR$(K%);:IFX<XHI%
-32 X=X+32 ELSEIFY>YLO%+40 Y=Y-40:X=XLO%
  920 ENDPROC
  930
  940 DEF PROCScrub:PROCDelete:IFX>=XLO%+32
X=X-32 ELSEIFY<=YHI%-40 Y=Y+40:X=XHI%-32
  950 PROCDelete:ENDPROC
  960
  970 DEFPROCBars:GCOL0,B%(K%-130,1):K%=B%(
K%-130,0):PROCLetter:GCOL0,3:ENDPROC
  980
  990 DEF PROCScreen
 1000 A$=GET$
 1010 IFA$="P" PROCDump:ENDPROC
 1020 IFA$="D" GOTO1040
 1030 ENDPROC
 1040 *OPT1,0
 1050 *SAVE "SCREEN" 3000+4F00
 1060 *OPT1,2
 1070 ENDPROC
 1080
 1090 DEF PROCDump
 1100  REM <<Your printer dump routine goes
here>>
 2000 ENDPROC
```

MINI TEXT-EDITOR UPDATE

by Tim Powys-Lybbe

Many members have written to say that they find the mini text editor (BEEBUG vol.1 no.7 p.31) extremely useful as a simple word-processor substitute. In the notes below we give ideas for further improvement of the editor, and its modification to suit different printers.

WORD COUNT

The following amendment ensures that the word count routine is not confused by two consecutive spaces:

```
 31042 IF ?K%=32 AND ?(K%-1)<>32
 THEN W%=W%+1
```

(leave in all spaces in this line).

PAGE FEED

The following line gives you a page feed at the end of your text on an Epson MX80III:

```
        31095 VDU 2,1,12,3
```

SINGLE LINE PRINTING

The printed program in BEEBUG vol.1 issue 7 gives triple spacing on some printers. To stop this you must delete lines 31044 and 31060.

NUMBERLESS PRINTING WITHOUT ASTERISKS

One problem with the text editor as it stands is that you must avoid certain characters that the Basic interpreter will recognise and modify. A way around this is automatically to insert an asterisk at the start of each line, and then ignore it in the printout routine. The following changes achieve this:

```
 25110 *KEY1 DELETE |M0@,1,100|M AUTO
 1100|M*
 25120 *KEY2 LIST 1000,1099|M AUTO 11
 00|M*
 25190 *KEY9 |M*
 31037 FOR K%=J%+2 TO M%*?J%-1
```

Then use key f9 instead of RETURN during the entering of text; if you forget and use RETURN then put an * as the first character - it will not be printed.

DISC VERSION

As we said in BEEBUG vol.1 no.8 p.38, if you are using the text editor on a disc-based machine, you will need to alter line 31022 of the program to:

```
        31022 M% = PAGE+1
```

This change only affects the 'print without line numbers' routine, and does not alter its correct operation on a cassette based system.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

ILLEGAL PRINT TAB USE

M.A. Ridyard would like to point out (Special note for all the commercial programmers) that with the new operating system, the PRINT TAB statement does not operate after VDU 5 code has been implemented (as it did on OS 0.1). This is mentioned in the new User Guide. With OS 1.2 (I am one of the lucky ones) a number of games are no longer usable (this is the unlucky bit) and machine code programs are difficult or impossible to modify.

Program tested on O.S. 0-1 and 1-2

INVASION (16k)

by Alan Webster

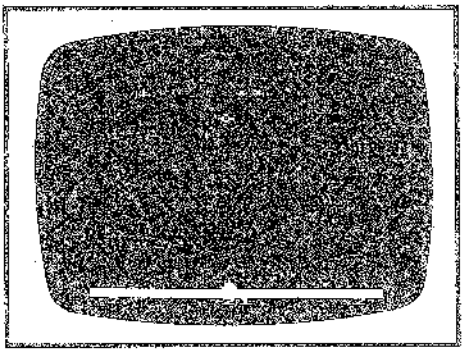
This is an invader type game for one player in which the object is to destroy all of the descending aliens (isn't it always?). The game starts with one alien attempting to 'land' on the surface of the planet. If this is killed then you move onto the next sheet with two aliens. This goes on until you have beaten off the sheet with six of the creatures, whereupon the game restarts with just one alien but becomes faster...and faster...and faster...!

You have three lives to play with and you lose one every time an alien lands. After losing all of your lives your score is displayed and the highest score is shown.

You control the cross on the screen, and have to move this directly on top of the alien and then fire.

The following keys are used within the game:

Z	Left	:	Up
X	Right	/	Down
	Space Bar to Fire		



```
10 MODE5
20 EK%=0:FR%=0
30 VDU19,1,11,0,0,0
40 PROCchr
50 HI%=0:DIM Z(5,2)
60 SC%=0:LV%=3:R%=10
70 FORTW%=0 TO 5
80 SOUND1,-10,235,1
90 OT%=TW%
100 B%=25
110 GF%=0
120 PX=9:PY=10:CLS
130 PROCscene
140 I%=1
150 PROCstars(100)
160 PROCrndm
170 IFI%=1I%=2ELSEI%=1
180 PROCalmove
190 SOUND0,-(RND(5)+10),7,1
200 IFLV%<1 PROCendgame:GOTO60
210 FORLP%=1TOR%+(R%ANDFR%=0)
220 PROCplmove
230 COLOUR2
240 PRINTTAB(2,31);SC%;TAB(17,31);LV%;
250 COLOUR3
260 PROCfire
270 NEXTLP%
280 IFEK%=TW%+1 EK%=0:NEXTTW%
290 IF TW%=6 R%=R%-1:GOTO70 ELSE170
300 END
310 DEFPROCscene
320 VDU23;11,0;0;0;0
330 VDU23,255,255,255,255,255,255,255,255,255
,255
340 FORA%=1TO18:PRINTTAB(A%,30);CHR$255;N
EXT
350 VDU23,254,8,8,28,28,62,62,127,127
360 PRINTTAB(9,29);CHR$254
370 ENDPROC
380 DEFPROCstars(A%)
390 FORA1%=1TOA%:GCOL0,RND(3)
400 PLOT69,RND(1280)-1,RND(960)+123
410 NEXT:ENDPROC
420 DEFPROCchr
430 VDU23,224,24,24,24,36,231,231,36,24,24
440 VDU23,225,192,153,60,90,126,60,66,129
450 VDU23,226,129,90,60,90,126,60,66,36
460 ENDPROC
470 DEFPROCrndm
480 FORA%=0TOTW%:Z(A%,0)=RND(18):Z(A%,1)=
0:Z(A%,2)=1:NEXT
490 ENDPROC
500 DEFPROCplmove
510 OX=PX:OY=PY
520 IFINKEY-98ANDPX>1GF%=GF%-1
530 IFINKEY-67ANDPX<18GF%=GF%+1
540 IF ABS(GF%)=2 PX=PX+SGN(GF%):GF%=0
550 IFINKEY-105ANDPY<27PY=PY+1
560 IFINKEY-73ANDPY>0PY=PY-1
570 PRINTTAB(OX,OY);" ":COLOUR3:PRINTTAB(
PX,PY)CHR$224
580 ENDPROC
590 DEFPROCalmove
600 FORA%=0TOTW%
610 IFZ(A%,2)=0 GOTO 700
620 PRINTTAB(Z(A%,0),Z(A%,1));SPC1
630 Z(A%,1)=Z(A%,1)+1:Z(A%,0)=Z(A%,0)+(RN
D(3)-2)
640 G%=Z(A%,0)
650 IFG%<1Z(A%,0)=1
660 IFG%>18Z(A%,0)=18
670 IFZ(A%,1)=29 AND G%=3ORG%=9ORG%=16 Z(
A%,0)=Z(A%,0)+1
```

```
  680 COLOUR1:PRINTTAB(Z(A%,0),Z(A%,1))CHR$
(224+I%)
  690 IFZ(A%,1)=29 Z(A%,2)=0:LV%=LV%-1:EK%=
EK%+1:FORSD%=1TO2:SOUND1,-15,5,4:SOUND1,-15
,1,4:NEXT
  700 NEXT
  710 ENDPROC
  720 DEFPROCendgame
  730 FORA%=1TO7000:NEXT
  740 CLS
  750 PRINTTAB(3,15);"YOUR SCORE:";SC%
  760 IFSC%>HI%HI%=SC%
  770 PRINTTAB(3,17);"HI SCORE  :";HI%
  780 *FX15,0
  790 COLOUR3:EK%=0
  800 S=GET
  810 ENDPROC
  820 DEFPROCfire
  830 FR%=0
  840 IFINKEY-99ANDB%>0FR%=608
  850 IFFR%=0 ENDPROC
  860 FX=64*PX+32:FY=1024-(32*PY)-16
  870 FR1%=96
  880 MOVEFR%,FR1%:DRAWFX,FY:GCOL0,0
  890 FORY%=0TOTW%:IFPX=Z(Y%,0)ANDPY=Z(Y%,1
)ANDZ(Y%,2)<>0 Z(Y%,2)=0:SC%=SC%+11-R%:EK%=
EK%+1:SOUND1,-15,100,1:SOUND1,-14,2,1:SOUND
1,-15,200,1 ELSE SOUND1,-10,235,1
  900 NEXT
  910 MOVEFR%,FR1%:DRAWFX,FY:GCOL0,3
  920 B%=B%-1:PRINTTAB(9,31);B%;CHR$32;
  930 ENDPROC
```

INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO

USER GUIDE ERRORS

The new User Guide has relatively few errors - to Acorn's credit. In BEEBUG vol.1 no.5 p.4 and no.7 p.22 we published a number which have been brought to our notice. Here are two or three more which have been recently discovered. We are grateful to Richard Florance, Andrew Smith and M. Elliott for pointing them out.

1) On page 452 of the User Guide, it states that when using OSFIND to open a file, the channel number is returned in the Y register. It is not, it is returned in the Accumulator (on version OS 0.1 anyway).

2) On page 179 the Basic line 10120 of the Moon-Lander program should read:

```
10120  MOVE X%,Y%
```

The User Guide has the X and the % sign around the wrong way.

3) There is a lack of information in the User Guide on the OSBYTE call with A=&81 (read key within time limit). Although the guide states (page 430) that the call can also be used to determine whether a particular key is being pressed - ie. like a negative INKEY - simply setting the Y register to the required negative number does not seem to work. However if both the X and Y registers are set to the negative number before the call is made, then on return they will both contain &FF if the key was pressed, or 00 if it was not.

A REQUEST FROM ACORN

Acorn are about to revise the User Guide, and they have asked us if we would put out an appeal for notification of errors. If you have discovered errors in the User Guide which have not yet been published, or if you have important suggestions to make regarding the Guide, would you jot them down and send them to the following address: USER GUIDE ERRORS, BEEBUG, PO BOX 50, ST.ALBANS. Please do not include other material, since all mail will be passed directly to Acorn.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

ON ERROR OFF

Peter Mabey suggests that readers should be reminded that when writing an error trap routine the first statement in the routine should turn off error trapping, or else any mistake therein will cause it to loop back to the beginning, so that the only way out would be to BREAK. i.e.:

```
10 ON ERROR GOTO 9999          9999 ON ERROR OFF
.  (main program)                .    (error routine)
```

SOFTWARE UPDATE - SOFTWARE UPDATE - SOFTWARE UPDATE

WORDWISE

WORDWISE AND FUNCTION KEYS

After using Wordwise and then returning to BASIC you may have noticed that the function keys cannot be programmed and any previous definitions are lost, the keys returning ASCII values from 160 onwards. Richard Bean informs us that the keys may be returned to their normal state, (with any previously entered definitions intact) by typing: *FX 225,1

EDITING BASIC PROGRAMS USING WORDWISE

George H. Foot has written in with a useful tip on editing a 'corrupted' Basic program using Wordwise. The principles could also be applied to general editing of a 'good' program using Wordwise, eg. using the 'Search and Replace' facility to change variable names etc.

Some unexplained corruption of data resulted in two lines 30 in one of my programs - one line 30 in the correct place, the other in the middle of the program. No action (even deleting that section of program) would remove the offending line 30. The solution was to transfer the program to WORDWISE and to use the word processing facilities to remove the unwanted line. It is a feature of WORDWISE that it can be used to edit BASIC programs and this can be very valuable.

The steps of the method are:-

Working in Basic, load the program.
*SPOOL "filename"
LIST "filename"
*SPOOL
Type *W. to transfer to WORDWISE
Load the SPOOLed file using Option 2 of Wordwise
Edit the program as required using the Wordwise editing facilities
Save the edited program using Option 1 of Wordwise (do not use option 8)
Return to Basic by typing *B.
*EXEC "filename".

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MEMORY SAVING ON ARRAYS

If your program has an array that stores 'byte-size' integers, ie. with a maximum value of 255, you can save memory by using the DIM A 100 technique. This allows you to write to the array (in this case called A and 101 bytes long) with A?x=...; where x is the array index, and to read it with Y=A?x...(Note, this is TUBE compatible). A further advantage of this method is that you can *SAVE/LOAD the result as follows:

```
10  DIM A 100
20  A$=STR$ A:B$=STR$ (A+100):REM This
    puts the address of A in Hex into
    A$, and the address of the finish
    of the array into B$
30  REM ... rest of program
100 DIM C 30
110 INPUT"File name "N$:IFN$=""
    OR LENN$)7 THEN VDU7:GOTO110
120 $C="SAVE"+CHR$34+N$+CHR$34+A$+CHR$
    32(acts as separater)B$
130 X%=C MOD256:Y%=C DIV256:CALL&FFF7
```

This technique allows you to SAVE & LOAD data at the same speed as normal SAVE/LOAD operations rather than the slower OPENOUT method. Richard C. Bean.

BEEBUG NEW ROM OFFER

ACORN PRESS RELEASE TO BEEBUG MEMBERS

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

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

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

NOTES ON ORDERING

1. To get a new ROM, BEEBUG members should send a cheque for £5.85 to ROM Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD. It is ESSENTIAL to include a cheque with order, and to give your membership number.
2. ROM orders must not be combined with any other order - eg for software etc; and because of constraints on supply, multiple orders cannot be accepted until further notice.
3. Because of uncertainties in supply, please allow 4-6 weeks for delivery. We undertake not to cash cheques until the week prior to despatch; and we will provide a monthly account of the supply situation in BEEBUG. Please keep a note of the date on which you posted your order so that you can relate this to future announcements.
4. Please note that we cannot accept EPROM-based 0.1 operating systems in lieu of payment. The exchange of EPROMs for the new operating system can only be performed by Acorn dealers or by Acorn's service centre at Feltham.

ADDRESS: ROM Offer, BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD.

WORDWISE Word Processor

BEEBUG Discount 13% SAVE £5

This is a highly sophisticated word processing package for the BBC Micro, and compares very favourably with those currently available on other microcomputers. It makes full use of the BBC micro's advanced facilities, and text is typed and edited in the 40 column Teletext mode, saving memory, thus allowing it to be used with more or less any TV. See the software review in this issue for further details.

Wordwise is supplied in EPROM with simple fitting instructions, a full manual, and a sample data cassette. Wordwise must be used in conjunction with a series 1 operating system.

The normal price of Wordwise is £39+VAT=£44.85

To BEEBUG members it is

EXTRA-SPECIAL OFFER

WORDWISE PLUS 1.2 ROM

AT £45.00 INCLUDING P&P & VAT.

WORDWISE PACKAGE PLUS NEW 1.2 ROM IS OFFERED AVAILABLE TO MEMBERS ONLY. THIS PRICE APPLIES TO THE UK ONLY. PRICE OUTSIDE UK IS £50 INCLUDING P&P.

MAKE CHEQUES PAYABLE TO 'BEEBUG' AND SEND TO: WORDWISE OFFER, BEEBUG, PO BOX 50, ST ALBANS. IT IS ESSENTIAL TO QUOTE YOUR MEMBERSHIP NUMBER WITH ORDER, AND PLEASE ALLOW UP TO 28 DAYS FOR DELIVERY ON THIS DOUBLE OFFER.

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 BEEBUG supplements are not supplied with back issues.

Subscriptions Address
BEEBUG
Dept 1
374 Wandsworth Rd
London
SW8 4TE

SUBSCRIPTIONS

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

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

SOFTWARE AND ROM OFFER (Members only)

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

Software Address
BEEBUG
PO BOX 109
Baker Street
High Wycombe
Bucks
HP11 2TD

IDEAS, HINTS & TIPS, PROGRAMS, AND LONGER ARTICLES

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

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

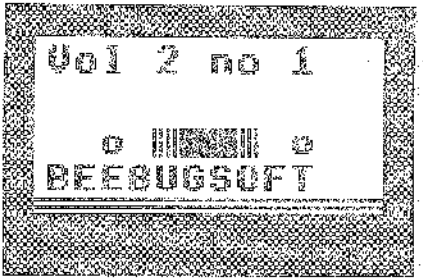
We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

BEEBUG NEWSLETTER is edited and produced by Dr David Graham and Sheridan Williams.
Technical Editor: Colin Opie. Production Editor: Phyllida Vanstone.
Technical Assistant: Alan Webster.
Thanks are due to John Yale, Adrian Calcraft, Tim Powys-Lybbe, and David Tall for assistance with this issue.

All reasonable precautions are taken by BEEBUG to ensure that the advice and data given to readers are reliable. We cannot, however, guarantee it, and we cannot accept legal responsibility for it, neither can we guarantee the products reviewed or advertised. BEEBUG (c) April 1983.

MAGAZINE CASSETTE OFFER

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



Vol 1 no 10
16 Programs including the following:

3D Rotation
Life
Artillery Duel
Square Dance
Disc Formatter
File Handler
Micro-Sketch
Date validation Routine

Vol 2 no 1
13 Programs including:

Four in a Row
Invasion
Music Composer
Bar Chart Generator
Basic Utility Editor
Double Height Routine
Disc Menu Program
PLUS FULL LISTING OF MINI TEXT ED VERSION 3, AND A PROGRAM TO PLAY BEETHOVEN'S BAGATELLE IN b FLAT (3 VOICES)

PRICE:Members UK Only £3.50 inc p&p and VAT
Europe £4.50 inc p&p (VAT not charged)
Elsewhere £5.00 inc p&p (VAT not charged)

Please give membership number. Non-Members add 50p per cassette.
If ordering more than one tape, deduct 20p from the price of each.
Make cheques payable to BEEBUG, and post to BEEBUG, PO Box 109, High Wycombe, Bucks, HP11 2TD.

BEEBUG BINDER OFFER

A hard-backed binder for BEEBUG magazine is now available. These binders are dark blue in colour with 'BEEBUG' in gold lettering on the spine. They allow you to use the whole of the first volume of the magazine as a single reference book. Individual issues may be easily added or removed.

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

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

Printed in England by Staples Printers St Albans Limited at The Priory Press.
ISSN 0263-7561