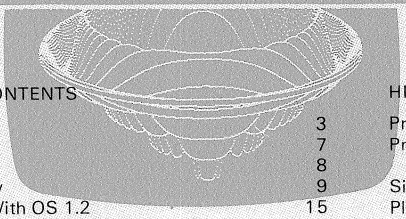
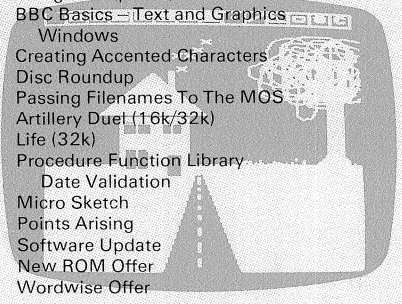
BBC MICRO USER MAGAZINE

BEEBUG

Vol. 1 No. 10 MARCH 1983

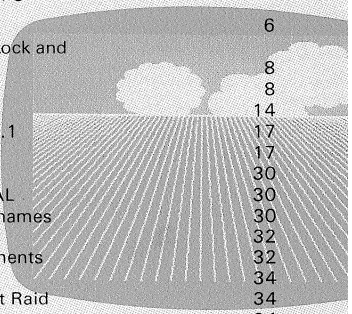


GENERAL CONTENTS



PROGRAMS

HINTS, TIPS & INFO



BRITAIN'S LARGEST MICRO USER GROUP

MEMBERSHIP NOW EXCEEDS 16,000

£1.00

EDITORIAL

1.2 ROM Deliveries.

There has been a predictably high demand by members for the 1.2 ROM, and we are starting to receive shipments of the device from Acorn. We are getting these out with a very short turn-around, but supplies inevitably lag behind orders - even though we have ordered many thousands of units from Acorn well in advance. Nevertheless we expect to be able to keep within the 4-6 week delivery period mentioned last month. If you do wait longer than this, we would urge you to read announcements in next month's magazine before writing to us, since it is almost certain that the delay will be beyond our control.

BEEBUG

This issue is again larger than the previous one, and we have tried to smarten up the cover a little. The next issue is our Anniversary issue, and will be accompanied by a BBC micro quick reference card. We hope to make it an even better issue than the present one; and it will also contain a cross-referenced index of all previous issues, and a binder offer.

NEXT MONTH – SPECIAL ISSUE

* Free BBC micro reference card with each issue.
* Utility Editor - a program which lists variables & procedures, and will search and replace variable names etc. within a program.
* Cassette Recorders for the Beeb - a review of currently available models.
* Composer - compose your own music with this useful utility.
* Torch discpack review.
* Implementing macros on the BBC assembler.
* Acornsoft Games reviewed.
* Multi-key - a program which effectively gives you up to 27 user defined keys.
* New 16k invader game.
* Bar Chart generator program.
* Double height character routine for all modes.
* Plus newcomers section and many hints and tips.

NOTICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BO

* Torch Review - If you are thinking of buying a Torch disc pack, it may be worth waiting for our review of this in the April 83 issue. We have some reservations about it.
* Software Savers. SEE IMPORTANT NOTE IN THIS MONTHS SUPPLEMENT.
* Hint Winners - This month's prizes of £10 and £5 for the best hints go to G Weston and Howard Spurr.
* Software Competition - This closes 15 April. Full details in this months supplement.
* MX80/100. We hear that Epson are about to launch a new range of printers. We will review these as soon as possible.
* Membership Number - Some members have been unable to find this - it is the number which appears on your magazine envelope label (for example Y62196).
* BEEBUGSOFT New software for members: EXMON and ARTIST2 - see page 44 for details.

Front cover illustration depicts screen shots from previous issues.

3D ROTATION (32k)

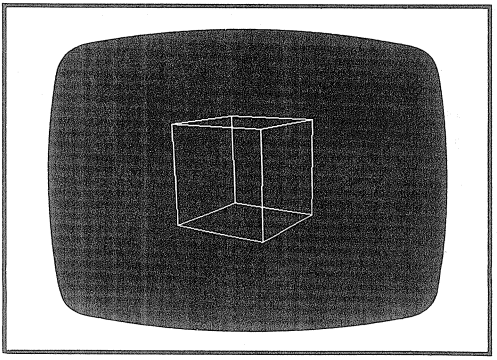
Text by Adrian Calcraft, Program by J Hastings

This is an extremely interesting program sent in by James Hastings which enables shapes to be drawn in three dimensions and then viewed from any angle, and at any distance.

The object to be drawn is defined in the program through data statements, which are easily set up, as they are the X, Y and Z co-ordinates of the "corners" of the object. Once these are entered the program is run, and displays a front view using mode 4 graphics. The object may then be rotated about any axis or viewed from nearer or further simply by pressing one of 8 keys. The program will only draw and rotate objects made up from straight lines, but there is no apparent limit to the complexity of the "wire-frame" objects generated.

HOW TO USE THE PROGRAM.

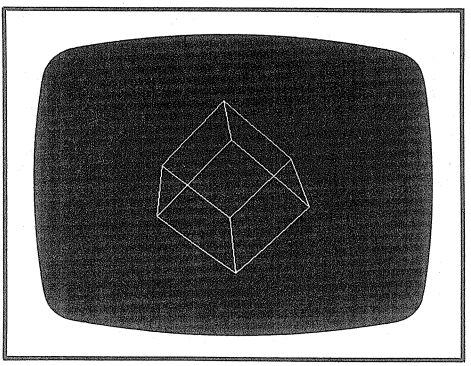
Type in the program as listed. The reason that the main program starts from line 1000 is to allow the earlier lines to be used for the data statements to define objects that you may develop later. The data statements on lines 30 to 80 define a 3D cube. Pictures of this are included with the article.



Run the program. You should see a cube, viewed from one side. The cube may now be moved using the following keys;

a) Cursor keys LEFT and RIGHT rotate around the Y axis.
b) Cursor keys UP and DOWN rotate around the X axis.
c) "RETURN" and the "]" key (left of RETURN) rotate around the Z axis.
d) "DELETE" and "COPY" reduce and increase the size of the object.

This provides the ability to view from any point, including actually from within the object itself. The program always draws all lines of the object concerned as if it were a wire frame.



CREATING A NEW 2-D OBJECT

Designing your own shapes is very easy. To start with let us consider how to draw a simple square in 2 dimensions. (The program will equally well draw and display 2D objects in a 3D field of vision).

Take a piece of paper (graph paper is even better) and draw a large cross in the middle. This will represent the X and Y axes upon which we will sketch the object, in this case a square. Now draw a large square with the cross at its centre. Number the corners of the square anti-clockwise, from 1 to 4, starting

with the top right corner as 1.

We now need to work out the X, Y and Z co-ordinates of the square. Let's assume for convenience that the length of each side of the square is 1000 units.

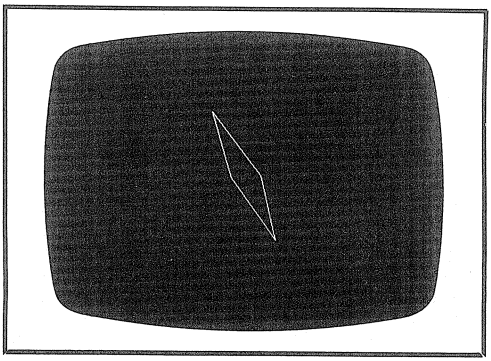
This makes the X and Y co-ordinates of point 1... 500,500. As the object is flat (we are only drawing it in 2D), the Z co-ordinate will be 0.
So the co-ordinates of point 1 are 500,500,0 (co-ordinates are always given in the order X, Y, Z)
Similarly those of point 2 are -500,500,0
Point 3's are -500,-500,0
Point 4's are 500,-500,0

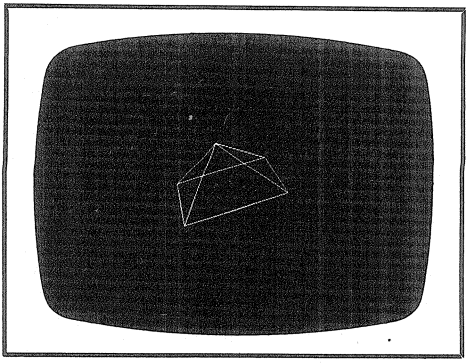
We can now compose the data statements for the program. These will be inserted into the program on any line numbers up to 990 and will replace those in the program listed below on lines up to 990. The program requires information in the following format.

1) Number of "corners", followed by the X, Y and Z co-ordinate of each "corner".
2) Number of lines to be drawn, followed by the "corner numbers" which they should join.

This may sound complicated but is in fact very straight forward. The number of corners in our square is obviously 4 and we have worked out the co-ordinates already, so the first data statement will read:

```
30 DATA 4,500,500,0,-500,500,
   0,-500,-500,0,500,-500,0
```





The number of lines is also 4. If you look at your sketch you will see that we have numbered the corners from 1 to 4. The lines of the square go from point 1 to point 2, point 2 to point 3, point 3 to point 4, and point 4 to point 1. This is all that is required for the second data statement, which we can now write:

```
40 DATA 4,1,2,2,3,3,4,4,1
```

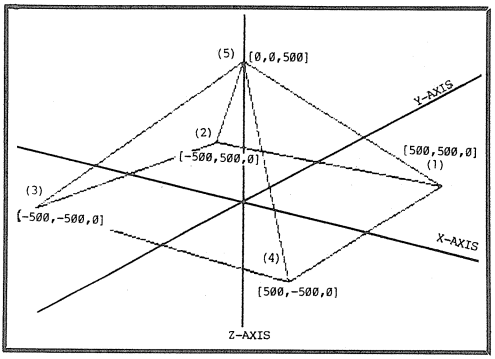
That's all there is to it. These two lines, 30 and 40, should replace the data statements in the program listed below on lines 30, 40 and 80. The actual line numbers are irrelevant as long as they are below 1000. Type them in and run the program, remembering to check that previous data statements, such as line 80 have been removed.

HOW TO DRAW YOUR OWN 3D OBJECT.

To do this we follow exactly the same process as above but now using an object with depth, eg a pyramid. As this is drawn as an extension to a square, we will be able to use some of the above calculated co-ordinates.

Take the sketch of the square made earlier and consider the Z axis. This is at right angles to the other two axes and can be thought of as extending from above the paper, through the centre of the cross, to below the piece of paper. When calculating 3D co-ordinates you have the choice of either imagining the points not actually on the paper, or attempting to sketch them using perspective.

To continue with the pyramid, consider its apex, which we will define, for convenience, at a height of 500 units. This places it exactly over the intersection of the X and Y axes at a height of 500. Consequently its co-ordinates will be 0,0,500. If we number the apex as "corner" 5, you can see that it will require lines to be drawn joining it to "corners" 1,2,3 and 4. We now have an object with 5 "corners" and 8 lines. The data statements can be therefore represented as follows:




```
30 DATA 5,500,500,0,-500,500,0,
   -500,-500,0,500,-500,0,0,0,500
40 DATA 8,1,2,2,3,3,4,4,1,5,1,
   5,2,5,3,5,4
```

Type in these lines instead of all other lines up to 990 and run the program.

If this doesn't make sense to you, compare them with the data statements calculated above for the square. Refer also to the picture of the pyramid accompanying this article, to see how the axes would pass through it. The co-ordinates of the "corners" are also indicated.

FORMULAE

For those people interested in the calculations used to perform the rotation, James has given the following formula. To calculate the new X and Y co-ordinates following a rotation about the Z axis by an angle of psi:

```
Xnew=Xold*Cos(psi)-Yold*Sin(psi)
Ynew=Yold*Cos(psi)+Xold*Sin(psi)
```

and so on for the other axes.

```
10 REM Points data
20
30 DATA 8,-500,500,500,500,500,500,
500,-500,500,-500,-500,500
40 DATA -500,500,-500,500,500,-500,
500,-500,-500,-500,-500,-500
50
60 REM Lines data
70
80 DATA 12,1,2,2,3,3,4,4,1,1,5,2,6,
3,7,4,8,5,6,6,7,7,8,8,5
90
1000 ON ERROR GOTO 1180
1010 MODE 4
1020 VDU 29,640;512;23;8202;0;0;0;
1030 *FX 4,1
1040 PROCassemble_CLG
1050 PROCpoints
1060 PROClines
1070 distance%=5000
1080 diststep%=500
1090 anglestep=PI/16
1100 REPEAT
1110 PROC_2D
1120 PROCdraw
1130 PROCupdate
1140 PROCrotate
1150 UNTIL FALSE
1160 END
1170
1180 REM Error trap
1190 MODE 7
1200 IF ERR<>17 REPORT: PRINT " at line ";
ERL
1210 *FX 4,0
1220 END
1230
1240 DEF PROCassemble_CLG
1250 REM Fast CLG routine
1260 DIM P% 25
1270 [ OPT 2
1280 ._CLG LDA #0
1290 LDX #0
1300 LDY #40
1310 .loop STA &5800,X
1320 INX
1330 BNE loop
1340 INC loop+2
1350 DEY
1360 BNE loop
1370 LDA #&58
1380 STA loop+2
1390 RTS
1400 ]
```

```
1410 ENDPROC
1420
1430 DEF PROCpoints
1440 REM Dimension point arrays and r
ead in points data
1450 READ points%
1460 DIM X(points%),Y(points%),Z(poin
ts%),X2D(points%),Y2D(points%)
1470 FOR count%=1 TO points%
1480 READ X(count%),Y(count%),Z(count
%)
1490 NEXT count%
1500 ENDPROC
1510
1520 DEF PROClines
1530 REM Dimension line arrays and re
ad in lines data
1540 READ lines%
1550 DIM start%(lines%),end%(lines%)
1560 FOR count%=1 TO lines%
1570 READ start%(count%),end%(count%)
1580 NEXT count%
1590 ENDPROC
1600
1610 DEF PROC_2D
1620 REM Convert to 2-D
1630 FOR count%=1 TO points%
1640 X2D(count%)=X(count%)*2500/(dist
ance%-Z(count%))
1650 Y2D(count%)=Y(count%)*2500/(dist
ance%-Z(count%))
1660 NEXT count%
1670 ENDPROC
1680
1690 DEF PROCdraw
1700 CALL_CLG
1710 FOR count%=1 TO lines%
1720 MOVE X2D(start%(count%)),Y2D(sta
rt%(count%))
1730 DRAW X2D(end%(count%)),Y2D(end%(
count%))
1740 NEXT count%
1750 ENDPROC
1760
1770 DEF PROCupdate
1780 phi=0: theta=0: psi=0
1790 REPEAT
1800 *FX 15,0
1810 key=GET
1820 UNTIL key=13 OR key=93 OR key=12
7 OR key=135 OR key=136 OR key=137 OR
key=138 OR key=139
1830 REM Rotate about X axis ?
1840 IF key=138 THEN phi=anglestep
1850 IF key=139 THEN phi=-anglestep
1860 REM Rotate about Y axis ?
1870 IF key=136 THEN theta=anglestep
1880 IF key=137 THEN theta=-anglestep
1890 REM Rotate about Z axis ?
1900 IF key=13 THEN psi=-anglestep
1910 IF key=93 THEN psi=anglestep
1920 REM Change viewing distance ?
1930 IF key=127 THEN distance%=distan
ce%+diststep%
1940 IF key=135 THEN distance%=distan
ce%-diststep%
1950 ENDPROC
1960
1970 DEF PROCrotate
1980 IF phi<>0 THEN PROCXrotation
1990 IF theta<>0 THEN PROCYrotation
2000 IF psi<>0 THEN PROCZrotation
2010 ENDPROC
2020
2030 DEF PROCXrotation
2040 REM Rotate about X axis
2050 Cosphi=COS(phi): Sinphi=SIN(phi)
2060 FOR count%=1 TO points%
2070 Y=Y(count%): Z=Z(count%)
2080 Y(count%)=Y*Cosphi-Z*Sinphi
2090 Z(count%)=Z*Cosphi+Y*Sinphi
2100 NEXT count%
2110 ENDPROC
2120
2130 DEF PROCYrotation
2140 REM Rotate about Y axis
2150 Costheta=COS(theta): Sintheta=SI
N(theta)
2160 FOR count%=1 TO points%
2170 X=X(count%): Z=Z(count%)
2180 X(count%)=X*Costheta-Z*Sintheta
2190 Z(count%)=Z*Costheta+X*Sintheta
2200 NEXT count%
2210 ENDPROC
2220
2230 DEF PROCZrotation
2240 REM Rotate about Z axis
2250 Cospsi=COS(psi): Sinpsi=SIN(psi)
2260 FOR count%=1 TO points%
2270 X=X(count%): Y=Y(count%)
2280 X(count%)=X*Cospsi-Y*Sinpsi
2290 Y(count%)=Y*Cospsi+X*Sinpsi
2300 NEXT count%
2310 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

PROGRAM LENGTH

To find how long a program is when you have saved it to tape, print the decimal equivalent of the last four figure hex number that appears on the screen. Dylan Reisenberger.

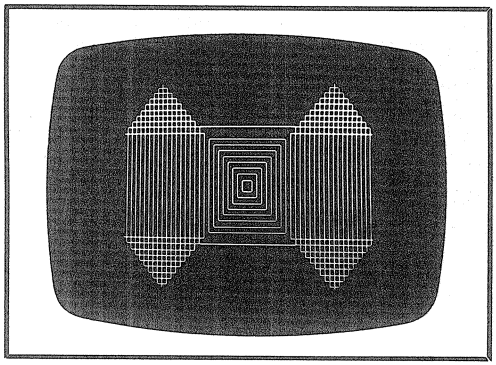
Program tested on O.S. 0.1 and 1.2

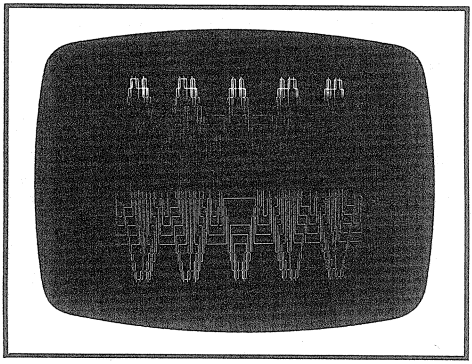
SQUARE DANCE (16k/32k)

by Martin Richards

This program displays a series of expanding and contracting rectangles which change colour during the course of execution. The actual 'form' of the patterns is random and therefore a number of runs will display different patterns. Some patterns repeat after a while, whilst others appear to keep on changing, giving a rather nice display of coloured graphics.

The commands GCOL and VDU19 (see BEEBUG no.9, p.3) are used to good effect in producing the many colours supplied. Some interesting fringe patterns are also produced if you watch the output on a monochrome TV, especially if you try it in MODE 1. If you have a 16K machine, you will need to alter line 20 to MODE 4.





PROGRAM ANALYSIS

```
*30        Gets rid of the blinking cursor (!)
*40        Puts the graphics origin at the centre of the screen
*50        Selects the initial height and width of the rectangle
*60        Selects the speed at which the sides are to change
*70        Selects the initial colour
*80 to 150 Main Loop (press ESCAPE to stop)
*90 to 110 Draws any particular rectangle
*120       Chooses the next width and height
*130       If rectangle is too wide then change direction and colour
*140       If rectangle is too high then change direction and logical colour
```

```
10 ON ERROR GOTO 160
20 MODE 1
30 VDU23;8202;0;0;0;
40 VDU29,640;510;
50 X=0:Y=0
60 DX=RND(50) :DY=RND(50)
70 C=RND(3)
80 REPEAT
90 GCOL 3,C
100 MOVEX,Y:DRAW-X,Y:DRAW-X,-Y
110 DRAWX,-Y:DRAWX,Y
120 X=X+DX:Y=Y+DY
130 IFABS(X)>640 THEN DX=-DX:C=RND(3)
140 IFABS(Y)>510 THEN DY=-DY:VDU 19,
RND(3),RND(7);0;
150 UNTIL TRUE=FALSE
160 MODE 7
170 REPORT :PRINT" @ line ";ERL
180 END
```

BRAIN TEASER

by Gareth Suggett

Gareth starts what we hope will be a long running series of puzzles for you to delve into. Many, like the one below lend themselves to a computer solution. There are no prizes because most of the puzzles will be investigative in nature, rather than having a unique correct solution.

PERSISTENCE

I am sure that you will have as much fun programming solutions as I have had, so here goes with the first of the series:

This puzzle originally appeared under this title in the now-defunct magazine "Games and Puzzles" in 1974. It is also discussed under the title "The pole of a number" in Rade and Kaufman's "Adventures with your pocket calculator".

Pick a 4-digit number. Generate a new 4-digit number as follows: Form the largest number that can be made from the digits of the original number and subtract from it the smallest such number. For example

4818 becomes 8841 - 1488 = 7353

Repeat the process on this new number:

7533 - 3357 = 4176

and so on:

7641 - 1467 = 6174

Further iterations continually reproduce 6174. Show that this same number is reached whichever 4-digit number is chosen initially.

For 5-digit numbers the process does not lead to a unique final number (pole) but ends in one of three cycles. Complete results are also known for 6 and 7-digit numbers. Find these and investigate the problem for numbers of 8 or more digits.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

PROGRAMMED CAPS-LOCK AND SHIFT-LOCK

Howard Spurr writes concerning software control of the CAPS-LOCK and SHIFT-LOCK functions and light-emitting diodes (LED's). Software control of these functions is required in such things as keyboard tutorial programs. The data is the same whatever OS is used, but the RAM address will differ.

For OS 0.1 the following will work:

```
?216=&10  Shift lock
?216=&20  Caps lock
?216=&30  Lower case
```

For OS 1.0 and 1.2 the relevant address is 602 (decimal). It is quite easy to scan the lower memory for an address that contains the above data in the appropriate modes.

If data is read at this address it will give the state of the SHIFT key (&08 added to above data), and the CTRL key (&40 added). Note that these routines will not work across the Tube when a second processor option is added.

SILENT GAMES

On the new series 1 operating system the 'effects' call *FX210,1 will turn the speaker off, so now you need not listen to all those crazy tunes and bangs that drive you insane. Note that you can switch the speaker on again by using *FX210,0. [There are many other goodies in the new 1.2 OS - see the relevant article elsewhere in this issue. Ed.]

PRINTERS FOR THE BBC MICRO

There are many printers available on the market and it is not always easy to know which one would most suit your requirements. In this review we take a look at four printers. In each case the reviewer has been using the printer for some time in conjunction with a BBC micro. The models concerned and respective reviewers are:

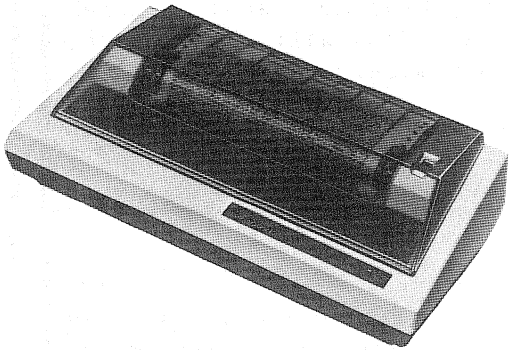
Printer	Type	Approx/Cost	Reviewer
a) SEIKOSHA GP-100A	Dot Matrix	£200	Adrian Calcraft
b) EPSON MX80/MX100	Dot Matrix	£320 / £430	Sheridan Williams
c) OLIVETTI PRAXIS 35	Daisywheel	£450	Graham Geatrix
d) TANDY CGP-115	Pen plotter	£149	Rob Pickering

The EPSON and SEIKOSHA models are matrix printers (ie. the characters are made up of a matrix of dots). The PRAXIS is a daisywheel printer giving true letter quality print but no graphics, and the TANDY CGP is really a pen-plotter which can generate text and graphics in four colours.

Even if you are primarily interested in just one of the given printers you may find it useful to glance over the other reviews. When purchasing a printer have a good look through the adverts - prices do vary and so does the service backup.

SEIKOSHA GP-100A

At a retail price of around £200, the GP-100A is one of the cheapest full size printers readily available for use with the BBC micro. Seikosha also make a smaller printer, the GP-80A, which is slightly cheaper, and have recently introduced the more expensive GP-250X.

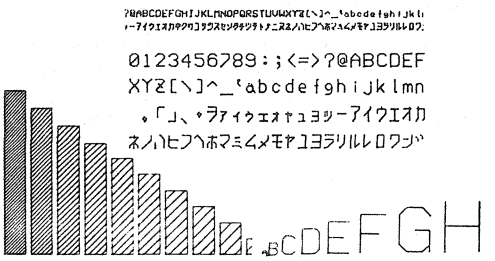


The BBC in conjunction with ACORN are selling the 80A and 100A under their own name. These are identical to the Seikosha printers.

The 100A uses a "tractor feed", which means that special paper with holes along the edges is required. This is no problem to get hold of and is relatively cheap. It does mean however that standard A4 sheets cannot be used, unless an add-on friction feed unit is also purchased. The paper holders on the printer are movable enabling the use of varying widths of paper from 4.5 to 10 inches. The listing paper must be stored directly behind the printer to ensure correct feed. This does not allow the paper to be trailed over the table to a box out of the way on the floor. Having observed this I have only encountered one or two misfeeds in some five months of using the printer.

The printing method used is 'impact dot matrix' producing characters on a 5x7 matrix, the actual size of which is 2.11 x 2.82 mm. The characters appear clear presenting no readability problems. The character set, consisting of upper and lower case characters, numerals and symbols totals 116. The maximum number of character columns is 80, the spacing being 10 chars. per inch. Unfortunately the 100A does not print "descenders", ie. lower-case letters such as 'p' 'g' and 'q' are raised slightly above the print line to enable the "tail" of the letter to be included, (see lower case printout).

Tandy CGP-115



During the course of
increasingly be seein
available in EPROMs and
basic production costs f
general awareness increa
programmer is an essential
anyone wishing to pu
information into EPROM,
ken a look at the cur
e market. Two companies
pplied machines for re
mpany failed to supply o
:*NOTE: If you want to
'ROMS or ROMS themselves

MATRIX OR DAISYWHEEL **Olivetti Praxis 35**

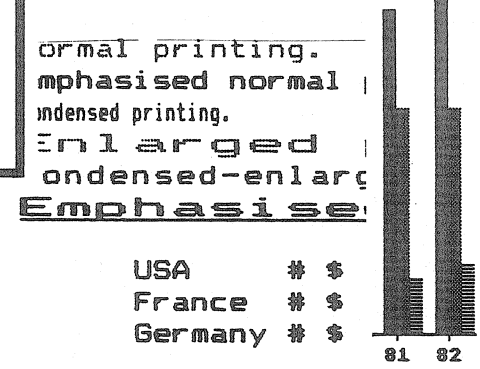
You can't afford more than £
print, high speed printout and
can't have all these featu

Matrix printers have every
quality print. Daisywheel pr
slow, have no graphics capab
sets. You will have to dec
compromise.

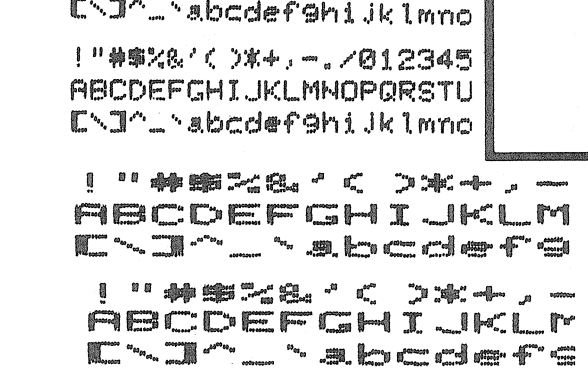
If LISTing is to be the
matrix printer is the one

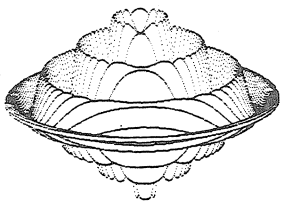
Epson MX80/MX100

enlarged'.
Most of the options a
elect 'emphasised enlar
f 8 different character



Seikosha GP-100A





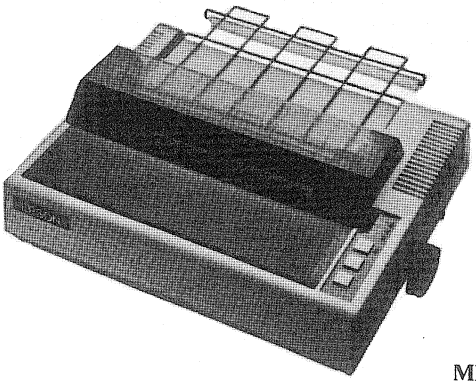
A switch on the machine enables the choice of one of four country character sets. This option allows the printing of special characters such as the German umlaut. The four settings available are USA, UK, Germany and Sweden. Double width characters are available and are enabled with the VDU 1,14 instruction. A sample of these is also given. The printer is quite noisy although not uncomfortably so, the plastic dust cover also acting as an effective baffle. The printing speed is 30 characters per second (cps), printing taking place only from left to right.

Graphics printing using special routines, such as the one published last month, produces good definition, with up to 480 columns across a sheet. The printer uses the parallel Centronics interface, and can be used by model B s or model A s with the printer upgrade. A serial interface may be purchased if desired.

As a tool for programming, a printer is invaluable. The 100A supplies many of the features of more expensive machines and produces good legible copy in both character and graphics modes. It is easy to use, relatively cheap, and I have had no problems with it so far. Having said this however, I would have reservations about using the printer to produce, say, documents to use in a commercial environment.

EPSON MX80/MX100

The Epson range of printers seem to represent very good value for money. This is why they are so popular amongst hobbyists, schools/colleges, and businesses.



MX80

Two of the range are reviewed here, the MX80-III and the MX100-III. They can be obtained from a variety of dealers and you could expect to pay approximately £320 and £430 respectively. Serial interfaces, with or without extended buffers, can easily be obtained, but cost extra. As standard the printer comes only with a Centronics interface. Machines can usually be supplied with a ready-wired cable for the Beeb at around £15 to £20 extra.

There is no need to review both printers in detail as they are very similar indeed. The MX100 has a wider carriage and will take paper up to 15.5 inches wide, as opposed to the standard 9.5 inches on the MX80. Also the MX100 has a print speed of 100 cps, as opposed to the MX80's 80 cps. If you buy the 'FT' models, rather than a simple 'T' type, then the feed can be either traction or friction operated, ie. 'FT' models will accept standard paper as well as holed listing paper.

The printers are relatively quiet, and steady in operation. You can even keep writing on the same table when the printer is printing. The EPSON print heads are very cheap and easy to replace when they wear out, which is NOT quickly. A plastic cartridge holds the ribbons and these are also easy to replace. In fact the printer is extremely well designed in terms of cost of maintenance and replacements.

The printed characters are of reasonably good quality with lower case descenders, and in these respects the Epson scores heavily over the SEIKOSHA reviewed above. Indeed, when printing in emphasised print with a new ribbon the quality is very good indeed. The formation of the characters is done with a 9-wire matrix head.

There are three main character sizes, known as 'normal', 'condensed', and

'enlarged'. There are also 'emphasised', 'double' and 'underlining' modes. Most of the options are selectable at the same time, for instance you can select 'emphasised enlarged underlined' if you wish. In this way you can select any one of 8 different character fonts, as displayed at the end of these reviews. All selections are made in software - ie. by just sending an appropriate VDU code.

Next on the Epson's repertoire is the capability to select either 'Normal' or 'Dual-density' bit image mode. These are used to obtain graphics printouts. [Please see BEEBUG no.9, p.10 for a screen dump program to achieve this].

When used with the Wordwise word-processing package it is fairly simple to define keys to set up all the styles of printing. You can even select both the pound and the hash sign in the same document. Similarly you can select crossed and uncrossed zeros. Unfortunately when changing character sizes the line lengths become confused.

Finally there is also a group of built-in foreign character sets. These are either software or hardware selectable.

OLIVETTI PRAXIS 35 DAISYWHEEL

You can't afford more than £500. You would like letter quality print, high speed printout and graphics capability. Sadly, you can't have all these features, and must settle for a compromise. Matrix printers have every conceivable facility except letter quality print. Daisywheel printers produce perfect print but are slow, have no graphics capability and more limited character sets. You will have to decide on your priorities and purchase accordingly.

For under £450 you could buy a superb matrix printer like the Epson MX100 FT III. The print is good and there are many 'styles' but they still retain a 'building brick' structure. On the other hand, there is a vast array of characters available and any non-standard characters can be programmed from your computer.

Finding a daisywheel printer and lead for under £450 including the dreaded VAT is quite a problem. Kram Electronics produce a Centronics interface, including a BBC version, for the Olivetti Praxis 35 electronic daisywheel typewriter. It costs under £500 all in. Its print quality is equal to that of the more expensive dedicated daisywheels. A switch on the interface enables control of the printer to pass from the computer to the printer keyboard. There are clearly other suppliers of the Praxis, who may provide even better terms.

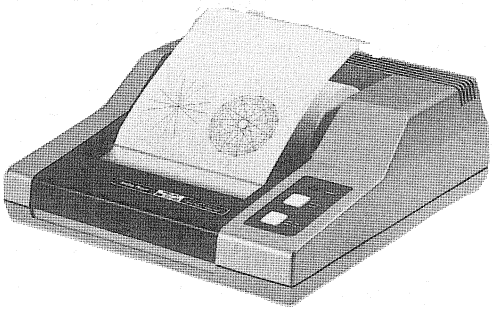
The printer doubles up as a high quality portable typewriter, and need not stand idle when the computer is switched off. It comes with two daisywheels, one with all the characters shown on the BBC keyboard except square brackets, curly brackets, up-arrow and tilde. Other wheels can be purchased in a variety of type faces including a special symbols wheel. It has a strong and neat carrying case but only one carbon ribbon cartridge. A carbon ribbon produces really superb quality print but costs £2 a time and lasts for 40000 characters or about 20 pages of closely packed A4 text. Nylon ribbons give ten times more print than the carbon variety. Each A4 sheet of print costs between a halfpence and one penny. The print quality is beautiful and vastly superior to anything that even the best matrix printer can produce.

The print speed is about eleven characters per second - about as fast as a good typist but less than half the speed of the MX80 on its slowest mode. An A4 sheet containing 3000 characters will take between six and seven minutes. A listing will take three minutes for a fifty line program.

The Kram interface is in a separate box measuring 12 x 20 x 6 cm. The printer has an area of 44 x 36 cm and varies in depth from 3 cm at the front to 12 cm at the back. It is a lot noisier than the Epson but a perspex cover can easily be made to cover the machine and this reduces noise to an acceptable level and keeps dust and little fingers at bay.

If your printer will spend most of its time producing listings, but letter quality print will be of paramount importance for all its other tasks, then the Praxis is a very good choice. There are rumours that the matrix printer will eventually spell the demise of the daisywheel, but this could be a long time coming. Other daisywheel machines will soon break the £500 barrier, but make sure that when a key is pressed on the BBC machine, the correct symbol is going to be produced by the printer.

TANDY CGP



The Tandy CGP (Colour Graphics Plotter), as its name suggests, differs from conventional printers. Unlike the daisywheel or dot-matrix printers it does not print from a ribbon, but actually draws the characters or graphics using miniature ball-point pens on to plain paper. Text and graphics can be plotted in up to four colours - black, blue, green, and red - and the plotting commands for the graphics are compatible with the BBC graphics commands.

The cost is £149 and it can be obtained from any TANDY outlet. If you can prove that you are a teacher (ie. in education) then there is also a discount awaiting you. It has two interfaces, a Centronics interface which is compatible with the BBC printer port, and a serial (600 baud) interface which is not compatible with the Beeb. It is a compact printer with dimensions of 210x216x75mm, weighing only 0.8kg. The complete package consists of the printer, a separate power supply, one roll of paper, 1 set of three black pens, 1 set of three coloured pens, and a user manual.

The main disadvantage of the plotter is the width of the paper, which at only about 115mm is about half the width of an A4 sheet. To offset this disadvantage I must point out that the maximum number of printed characters across the width is 80, and they're very readable too!

Controlling the printer is easy. You can list to it the same as any other printer when it is in text mode, but you may also place it into graphics mode by sending an ASCII code 18 to it. This is most easily done with VDU1,18. Whilst in graphics mode you may still print text, but in addition there are a large number of commands available to control movement of the pen.

All the commands start with a capital letter, some of which are followed by a number sent as characters - not as binary e.g. VDU2:PRINT"D100,200":VDU3 would 'Draw' a line from the current pen position to position 100,200. Very simple. By sending each character to the printer only (using VDU1,n) you avoid printing all the

commands on the screen at the same time as controlling the plotter. In this way, you can issue an instruction to the printer which corresponds to each PLOT, DRAW, or MOVE command drawing the same picture on the screen. You can also use the colours, though you only have four, and there are no flashing colours! The command: VDU2:PRINT"C1":VDU3 would select the colour blue, while "C2" would select green and so on.

Because there are so many commands I will not describe them all here. Suffice it to say that there are plenty to choose from, they are easy to use, and they provide many varied functions. You can even tell it to plot axes via a single instruction.

There are 480 plottable points in the x direction, and the y is unlimited, though effectively -999 to +999 from a current position. The paper moves either up or down with equal ease. Because of the width, a full BBC screen size is not quite achievable, though by swapping the order of x,y values in the commands, you get an actual size of 999 by 480. since the BEEB screen has 512 actual points in the y-direction at best, the loss is very rarely ever noticed.

A very good easy-to-follow manual is supplied with the printer, though all the test programs within it are designed to work on a Tandy computer. It is easy to convert these to BBC Basic by using a procedure to replace the Tandy Basic LPRINT command which sends output to the printer only. The manual does NOT give connections for the Beeb, but an absolutely standard BBC Parallel interface lead is all that is necessary for full operation.

Personally I would try to award this printer more than full marks; I haven't seen anything so good as this since I got a BBC micro. The paper is only just over £1 per roll, and having used this printer for heavy tests for hours on end for 6 weeks I am about 2/3 through the original roll of paper. Replacement pens are only £1.70 for three, and each of those lasts for 250 metres according to the manual, and I've used up two.

We are grateful to Mr Graham Pinder for initially introducing us to the TANDY CGP plotter.

STOP PRESS STOP PRESS

Information has just been received stating that KRAM Electronics Ltd., the supplier of the Praxis mentioned above, have gone into liquidation. There is, however, no shortage of suppliers of this machine. We recently contacted two, DATARITE TERMINALS Ltd of ESSEX and SIMMONS MAGEE COMPUTERS Ltd of TWICKENHAM, and both stock the Praxis together with a suitable interface.

STOP PRESS STOP PRESS

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

PLANETOID RESTART

Stelios Charalambides writes - "have you ever lost your game of Planetoid because you pressed the BREAK key, or for some other reason? Reloading is a pain because of the slow cassette system. The solution is very simple.

Once the program is loaded, or before you start loading, define a free key thus:

*KEY0VDU22,2,23;11;0;0;0;0:CA.&1C00|M

When the program becomes lost or crashes, and as long as the memory is not erased, you can restart the program by pressing the above defined key".

WHAT TO DO WITH 1·2

By David Graham
and M.E. Horton

The new 1.2 operating system is now available, either through BEEBUG - see this issue for details, or at Acorn dealers; and new machines are currently being supplied with 1.2 already installed. To find out what system you have, type *FXØ <return>. We have suggested that 1.2 is well worth having; and this month we look briefly at some of the features that it offers.

BUGS FIXED

Most importantly, the following bugs have been fixed:

1. The cassette filing system block zero bug.
2. The cassette data handling problem with PUT & GET byte.
3. The bugs in PLOT routines that cause odd things to happen on the bottom right of the screen have been fixed.
4. OS 1.2 will support "paged" ROMS - though if you have a Model A without an official upgrade, then see the article on 'Upgrading Model As to read paged ROMs in this issue.

ENHANCEMENTS

The 1.2 operating system carries a considerable number of enhancements, many of which are also available on 1.Ø.

1. New PLOT commands incorporating fill routines - PLOT instructions 72-79 and 88-95. See BEEBUG no. 6 p.1Ø for further details.
2. The ability to use the <shift> key in conjunction with the user keys to produce coloured and flashing text characters in mode 7, or with the <ctrl> key to produce coloured graphics characters in mode 7. E.g. press <shift> & function key 5 simultaneously then any other characters - they will appear in cyan. Press return then press <ctrl> and function key 3 simultaneously then any number or lower case letter, this will generate teletext graphics symbols.
3. A cold start is actuated by <ctrl> <break> rather than <break> <break> - ie. press <ctrl> and <break> simultaneously.
4. The introduction of a host of new *FX and OSBYTE calls with a variety of functions.

NEW *FX AND OSBYTE CALLS

Many of the new calls are documented in the User Guide, and on pages 418 and 419 a summary of FX calls is given with an indication as the whether they are new to series one (ie 1.Ø or 1.2 etc). In what follows, we will draw your attention to some of the more interesting of these, and then give a summary of SOME of the new undocumented calls.

DOCUMENTED *FX CALLS

*FX18 Reset user defined function keys. We put this in a Hint last month. The call clears the user defined key buffer. This is particularly useful, since redefining a previously defined key can give a 'keys full' message because the old key definition is not erased before the space check is made. You can of course empty each key individually with *KEYØ etc, but this is laborious.

*FX2Ø Explode soft character RAM allocation - this allows all characters to be redefined. See User Guide p427.

*FX21 Flush selected buffer. A useful call to enable queues in selected buffers to be deleted, including the keyboard buffer, the RS 423 buffer, the printer output buffer, the four sound channels and the speech synthesis buffer. To illustrate the use, it is a good idea to execute *FX21,Ø (flush keyboard buffer) before reading the keyboard with a GET, since any keyboard presses

before the GET would otherwise be stored in the buffer, and given out to the GET command. Alternatively use *FX21,0 at the end of a game, and before asking "ANOTHER GO ?" - otherwise the answer will be the last pressed key - eg "Z" for move laser base left etc.

*FX138 Insert character into keyboard buffer. See U.G. p.433 for syntax. By using this call you can make a program issue commands at the end of execution that are not normally allowed from within a program. E.g. the command NEW. The program below achieves this. If you run it, at the end of execution - ie after line 80 has been printed, NEW will appear on the screen, and the program will be erased. The data sent in the FX call are the ASCII codes for the required buffers (see U.G. p.486). The 13 is a carriage return character.

```
10 PRINT"TO EXECUTE NEW"        50 *FX138,0,69
20 PRINT"WITHIN A PROGRAM"      60 *FX138,0,87
30 PRINT"USING *FX 138"         70 *FX138,0,13
40 *FX138,0,78                  80 PRINT"THE PROGRAM HAS GONE"
```

*FX146 -151 Execute reads and writes to memory-mapped input and output devices - eg the video ULA, user port etc. Without these *FX calls, the only way to use the user port (see BEEBUG no.3 pp. 8-10) is to POKE and PEEK, which is undesirable because it means that such programs will not work across the Tube. See User Guide pp. 435-437 for further details.

*FX225 -228 Set the base number for function key codes. This can be extremely useful. As well as being able to store command strings and the like, the red function keys may also be used to generate a range of ASCII character codes. Good use has been made of this in that if you press <shift> and any function key simultaneously (not zero), you will generate an invisible Teletext code (if you are in mode 7), and any further characters typed will be either in colour or flashing. If you use <ctrl> and a function key, then the lower case letters will produce coloured graphics characters in mode 7. Try <ctrl> f3 (simultaneously) followed by "y" for example. Simultaneously pressing <ctrl> and <shift> and a function key produces a further range of codes. The *FX calls 225-228 allow you to change the range of codes generated. For example, if you execute *FX228,246, then <ctrl> <shift> f0 will put character 246 on the screen, f1 will give you 247, and so on up to f9 which will give 255. You can define each of these to produce whatever character you wish in all modes but 7, using the VDU23 call.

UNDOCUMENTED *FX CALLS

*FX196 Period before auto-repeat on keyboard starts (as set by *FX 11).

*FX197 Frequency of auto-repeat (as set by *FX 12).

*FX210 Cancel sound output. *FX210,1 turns off the sound generator, *FX210,0 turns it back on.

*FX219 Redefine function of Tab key. *FX219,49 sets it to ASCII character 49 (=1), *FX219,12 will make it clear the screen. *FX219,9 resets it to a normal tab function. It can be used in conjunction with user defined characters (see 'Accented Letters' in this issue).

*FX220 Read or set the ASCII character which gives 'Escape'. The default value is 27. If you execute *FX220,48, then pressing zero will execute an escape. *FX220,0 will disable the keyboard escape.

*FX236 Destination for output, as set by *FX 3.

*FX237 Cursor edit state (as set by *FX4).

*FX245 Printer output destination, as set by *FX 5.

*FX246 Printer ignore character, as set by *FX 6.

*FX247 -249 After the user has pressed <break>, the operating system looks at three locations in page two to find out what it should to. Normally, these bytes contain zeros, and the machine ignores them. Using these three calls, you can insert machine code instructions into these bytes. When "<break>" is pressed, the operaeing system will then jump to the first of the locations. As there are three bytes available, there is room to insert a JMP instruction. For example, to make the machine execute a routine which is at

address &DØØ after you press <break>, type:
*FX 247,76 (JMP instruction)
*FX 248,Ø (low byte of address)
*FX 249,13 (high byte of address)
If you want to return to normal OS break handling after your own routine has been executed, your routine should end with an RTS instruction. The effect of these calls is not reset by <control> n <break>.

*FX252 The number of the sideways ROM entered after <break>.

*FX253 Returns the type of the last reset to occur. X=Ø if <break> was pressed; X=1 if the last reset was power up; X=2 if <control> and <break> were pressed. (See also OSBYTE note at the end of this article).

*FX254 If used in read mode, returns &8Ø on a 32k machine and &4Ø on a 16k machine. In write mode, this provides a means to mimic a 16k machine on a mode B. Try *FX 254,&4Ø then press <control> and <break> together on a 32K machine. After this, you should get 'BBC Computer 16k'. It provides an easy way of making sure that your programs will run on both models. The call seems to work on a series 1.Ø S and is reset by *FX 254,&8Ø. Data of 255 and Ø seem to work just as well, and are easier to remember.

*FX255 Read or set startup options. The eight bits of this byte correspond to the links on the bottom right-hand side of the keyborad PCB which are used to select the screen mode entered after a reset, and to select some disc options. In write mode, you can use this call to change these options. For example, to make the machine enter mode 3 when you press <break>: *FX255,251 (251 - 11111Ø11 binary) and to get back to mode 7 after <break>: *FX 255,255 The last three bits of the second argument give the screen mode to be used. Again, this call works on OS 1.Ø. <control> <break> resets this byte to the value determined by the PCB links. The effect of the disc links was given in BEEBUG no.9 p.8. These can also be simulated with this call.

OSBYTE CALLS

All of the above *FX calls can be executed from machine code as an OSBYTE call. FX calls do not return a value (FX Ø is an exception), to do this the equivalent OSBYTE call must be made (see User Guide p.429). FX 253 is a read-only call, so there is little point in using this; OSBYTE 253 should be used to read the type of reset which last occurred.

INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO

PLOTTING BUG IN Ø.1 OS

If you try to draw a triangle whose apex is in line with its base you get a very odd effect. This bug was referred to by J Yale in his ARTIST program in the December/January issue of BEEBUG. The following will show the bug:

```
1Ø MODE 4                    4Ø DRAW 7ØØ,8ØØ
2Ø FOR I=1ØØ TO 7ØØ          5Ø PLOT 85,5Ø,8ØØ
3Ø MOVE I,8ØØ                6Ø NEXT
```

The bug does not exist in the series one OS.

Tim Powys-Lybbe.

BASIC ROM BUG

David Wright tells us about a bug in the BASIC ROM (issue one) which a friend of his uncovered. The routine for DIMensioning arrays does not check for 'silly' arguments. Hence DIM A (n*256-1,n*256-1,.....) produces some interesting results. For example DIM A (8191,8191) doesn't just crash the machine, it kills it stone dead - even the cursor vanishes. Some other (impossible) values are accepted, and you can print out the values in the array.

UPGRADING MODEL A s TO READ PAGED ROMs.

by Colin Opie

When you have a series one operating system (1.Ø or 1.2) your computer should be able to use the 'paged ROM' facility. This means that you can plug in ROM-based software (such as a word-processor or a disc-filing-system) into any of the three spare sockets 88, 1ØØ and 1Ø1 below the keyboard on the right hand side of the machine. However if you have a model A which has not received a full official upgrade it will probably need some hardware modifications to activate the paged ROM facility. The best way to check this in the first instance is to proceed as follows:

1. Unsuitable ROM Decoding -

A simple check for this condition is to put the Basic ROM in one of the three sockets 88, 1ØØ or 1Ø1 mentioned above, and see whether or not the computer is capable of recognising its existence. If it doesn't find Basic you will get the error message 'Language ROM?' when you switch the machine on.

This effect is most likely due to the machine being a model A computer (with or without 32k of RAM). The solution is to upgrade the ROM selection mechanism to the model B specification. This upgrade can be achieved by inserting IC76 (a 74LS163) and cutting links S12, S13. Both operations MUST be performed - under no circumstances should the machine be turned on with IC76 inserted and the links still connected. IC76 is situated just behind the keyboard socket (near pin 1, the left hand side of the socket) on the main pcb. The links S12, S13 are in a similar position, but near pin 17, the right hand side of the socket.

With the above modifications you should find that the various software ROMs (ie. Basic or Wordwise, but NOT the Operating System ROM) will be recognised in any of the three paged ROM sockets. It is important to note that the ROM in the right-most socket (the one up against the side of the case) is the one which will be selected on a cold-start. For example, there is no reason why you should not get your machine to always start-up in Wordwise, and only switch to Basic (via *BASIC) when you want to.

2. Wrong Link Positions -

There are a number of moveable links in the BBC computer which are pertinent to the task of ROM address decoding. It is vitally important that each of these links are set correctly.

In the following table 'North' means that the link joins the two pins (out of a set of three) that are furthest north, ie. nearer the back of the main board. 'East' follows the same convention and therefore refers to the two pins nearest the right-hand side of the board. 'West' is clearly the opposite condition to 'East', and 'East-West' refers to the general direction of the links (due to there only being 2 pins).

S18	north
S19	east
S2Ø	north
S21	2 x east-west
S22	north
S32	west
S33	west

NOTE If you are in any doubt about performing the above, or any other hardware modification, then consult your nearest dealer.

FILES (PART 2)

by Sheridan Williams

Sheridan Williams continues his series on using files, and presents a general file-handling program which will work on cassette or disc, though which is best suited to the former. In fact all comments referring to cassettes in this article also apply to discs.

In part one we concluded by saying that there were four solutions to the problem of updating a file. For those with a cassette player the best solution is the first one listed - load the file from cassette into an array, make all the modifications (including any additions) and then write the file back to cassette. This month I will discuss this solution and give an example program. This program will process the PEOPLE file whose record description we gave in part one. By process I mean that it will allow you to enter details about a particular person; you will be able to update that data; you will be able to save a copy of the file on cassette or disc. The program is listed at the end of the article.

SIMPLE FILE-HANDLING PROGRAM

The program keeps the data file to be managed in an array, so we must first declare the size of the array for each of the fields that we wish to use. The appropriate dimension statements are:-

```
10 nr=100
20 DIM sur$(nr),title$(nr),sex$(nr),dob$(nr)
```

The variable nr in line 10 holds the number of records that the file will have. This makes it relatively simple to change the value of nr at a later stage if we require to.

The whole program is built around the procedures PROCreadfile, PROCwritefile PROCaddrecords, and PROCprintcontents, and starts off (lines 80-300) by displaying a 'menu' of choices and deciding which procedures to use depending on that choice. Let us assume that you choose option 3 - add records to the file. This is the most sensible one to start with as initially the file will not exist, and option three allows it to be created and filled.

PROCaddrecords (lines 3000-3999) asks you which record number you are going to use (3040), if this is higher than any record used so far, it remembers it in the variable 'maxrec' (3045). It further checks that the record number is within acceptable limits (3050), and if the record already contains some data it will display it (3060) before asking you to enter the data (3070).

PROCreadfile	Read the file from disc/cassette into arrays in main store.
PROCwritefile	Write the file from the arrays in main store onto the cassette/disc file.
PROCaddrecords	Read records from the keyboard and put them into arrays in main store.
PROCprintcontents	Display on the screen the contents of the file held in arrays in main store.

Some of the procedures use a defined function 'FNfilename' this simply returns a filename that it asks you to enter.

PROCgetrecord uses a very useful technique, that if a 'null' field is entered it will hop back to the previous field. For example, suppose that you have just entered BLIGGS for the surname by mistake instead of BLOGGS, and you have pressed 'return', the program now requests "Title?". If you just press 'return' the program will hop back to request the surname again.

The variable 'maxrec' declared in line 30 (as maxrec=1) holds the highest record

actually used so far. It would be rather wasteful always to process all the 'nr' records that were dimensioned when perhaps most of them are not used.

It might be a good idea to spend some time studying the program. It is not complicated, but it uses statements with which you may be unfamiliar. If you wanted to extend the number of fields you would need to declare another array in line 20, as well as altering lines 2050, 4010, 5050, and extending PROCgetrecord appropriately. It would also be reasonably simple to add procedures for other routines such as searching and sorting.

The program as presented has a severe limitation - it can only process the PEOPLE file with about 900 records in a 32k machine or about 250 records in a 16k machine. If you add any more routines to the program this will have to be subtracted from the amount of memory available to hold the file. The solution is to use discs; and we will take a closer look at discs next month.

Errata:

[In part 1 I said that the last program would only read back 'Apple' from the file. This applies only to disc files because the operating system, sensing that the file has the same name when used in line 40 as when used in line 10, will use the same area on disc, hence overwriting the original data; whereas on cassette you can have several files all with the same name].

```
   10 nr=100
   20 DIM sur$(nr),title$(nr),sex$(nr
),dob$(nr)
   30 maxrec=1
   80 REPEAT:MODE 7
  100   PRINT TAB(5,3)"PEOPLE FILE PR
OCESSOR"
  120   PRINT''"1. Read a file."
  130   PRINT"2. Write a file."
  140   PRINT"3. Add records to the file."
  150   PRINT"4. Print contents of file."
  160   PRINT"5. End the program."
  200   PRINT''"CHOICE? ";:choice$=GET$
  210   IF choice$="1" PROCreadfile
  220   IF choice$="2" PROCwritefile
  230   IF choice$="3" PROCaddrecords
  240   IF choice$="4" PROCprintcontents
  300   UNTIL choice$="5"
  999 CLS:END
 1000 DEF FNfilename
 1010 REPEAT
 1020   INPUT''"Name of file (1-7 let
ters) ",filename$
 1030   UNTIL filename$>"" AND LEN(fi
lename$)<8
 1040 =filename$
 1999
 2000 DEF PROCreadfile
 2020 c=OPENIN(FNfilename)
 2030 rec=1
 2040 REPEAT
 2050   INPUT#c,sur$(rec),title$(rec)
,sex$(rec),dob$(rec)
 2055   IF rec>maxrec maxrec=rec
 2065   rec=rec+1:IF rec>nr rec=nr
 2070   UNTIL EOF#c
 2090 CLOSE#c:ENDPROC
 2999
 3000 DEF PROCaddrecords
 3030 REPEAT
 3040   INPUT''"Rec no (0 to return t
o menu)",rec
 3045   IF rec>maxrec maxrec=rec
 3050   UNTIL rec>=0 AND rec<=nr
 3055 IF rec=0 ENDPROC
 3060 IF sur$(rec)>"" PRINT"Record cu
rrently contains:":PROCprintrec
 3070 PROCgetrecord
 3120 GOTO 3030
 3999
 4000 DEF PROCprintrec
 4010 PRINTsur$(rec)" "title$(rec)" "
sex$(rec)" "dob$(rec)
 4020 ENDPROC
 4999
 5000 DEF PROCwritefile:LOCAL c,rec
 5030 c=OPENOUT(FNfilename)
 5040 FOR rec=1 TO maxrec
 5050   PRINT#c,sur$(rec),title$(rec)
,sex$(rec),dob$(rec)
 5060   NEXT rec
 5070 CLOSE#c:ENDPROC
 5999
 6000 DEF PROCprintcontents:LOCAL rec
 6005 CLS
 6010 FOR rec=1 TO maxrec
 6020   IF sur$(rec)>"" PRINT"Rec ";r
ec;":";:PROCprintrec
 6030   NEXT rec
 6040 INPUT"Press 'return' for menu", Q$
 6050 ENDPROC
 6999
 7000 DEF PROCgetrecord
 7010 INPUT''"Surname",sur$:IF sur$=""
 ENDPROC
 7020 INPUT"Title",title$:IF title$="
" THEN 7010
 7030 INPUT"Sex",sex$:IF sex$="" THEN
 7020
 7040 INPUT"Date of birth",dob$:IF do
b$="" THEN 7030
 7050 sur$(rec)=sur$:title$(rec)=titl
e$:sex$(rec)=sex$:dob$(rec)=dob$
 7060 ENDPROC
```

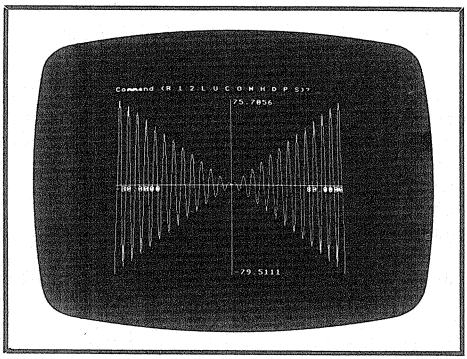
BBC BASICS – TEXT AND GRAPHICS WINDOWS

by David Graham

Text Windows

A text window is an area of screen where text may be printed. The default value of this on the BBC machine is the whole screen. This means that characters may normally be placed anywhere on the screen, by using the cursor keys or the TAB command; and when scrolling takes place, the whole screen scrolls.

It is possible to reset this window to a rectangle of any size or shape (in whole numbers of characters) and to place it anywhere on the screen. This can be particularly useful in graphics modes, when you may wish to display and scroll text while keeping a picture or diagram stationary on the screen. Such a technique is used in BEEBUGSOFT's Superplot, where a full width window of only one line in depth scrolls user information, and allows data input, while keeping the remainder of the screen for the display of the plotted functions.



The command that makes this possible is a VDU 28 call. It is documented on p. 387 of the User Guide, and its syntax is as follows:

VDU 28,leftX,bottomY,rightX,topY

As an example: VDU 28,19,31,39,0 would confine text to the right-hand half of the screen in mode 4. The scrolling window begins at character 19 from the left, and ends at no.39 (i.e. the rightmost position). The bottom line of the scrolling window is line 31, and the top line is line zero - hence the 4 parameters in the call. Try issuing this command (once you are in mode 4), then listing a program. It will scroll up the right-hand half of the screen. Note that the scrolling speed is much slower than usual because the clever fast-scroll technique used on the Beeb (hardware scrolling using the 6845 video controller chip) cannot be performed when windows are in use. If you want to create a similar window in mode 7, you will need to alter the 'bottom Y' figure from 31 to 24 (since there are only 25 lines in mode 7). It is worth noting that there is no error reporting on windows that overlap the screen; incorrect commands are just ignored.

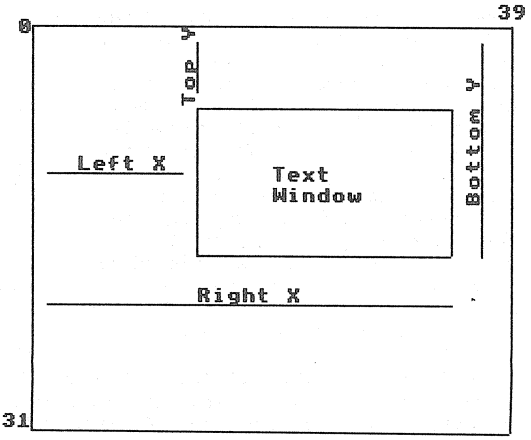


FIG 1

One clever use of the text window feature in mode 7 was given by Simon Wilkinson in BEEBUG no.6, p.30. It is worthy of reproducing here. The object of Simon's

program is to give permanently coloured text and background in mode 7. He achieves this by printing the appropriate control characters at the start of each of the 25 lines of the Teletext screen, and then shrinking the text window by three characters from the left so that the colour codes are not over-written, or scrolled away.

```
10 MODE7
20 N = 132 :REM Blue text
30 B = 134 :REM Cyan background
40 CLS
50 FOR Y = 0 TO 23
60 VDU B,157,N,13,10
70 NEXT Y
80 VDU 28,3,23,39,0
```

Another use of the text window is to allow small sections of text to remain on screen while the bulk of the screen is allowed to scroll. This technique is often used in word processors so that line length indicators and other information can be displayed against a scrolling screen of user-supplied text. The way to achieve this is to print the required permanent text at full window size, then reduce the text window so as to protect it. A similar technique could be used to provide permanent information about the contents of the user keys in say the bottom three screen lines. Again, if your objective is to combine small amounts of text with large graphical displays, then it is probably more useful to use scrolling windows of full screen width, but only two or three lines high for example. Such cases call for the combined use of text and graphics windows.

Graphics Windows

The VDU28 text window call has a counterpart in graphics. This is the VDU24 call (User Guide p. 385). It is used in a similar way to VDU28, except that its range of values corresponds to the graphics coordinates - (0 - 1023 vertically, and 0-1279 horizontally). The syntax is also broadly similar, but note the abundance of semicolons:

```
VDU 24,leftX;bottomY;rightX;topY;
```

Once you have defined a graphics window in this way, even if you issue PLOT & DRAW commands covering larger areas of screen, only those areas within the graphics window will be affected. There is a variety of uses for this command, one rather interesting application is to use it as a fast fill routine for rectangles of any shape or size. The principle behind the technique is to define a graphics window in the position of the required filled rectangle, then clear the graphics screen to the desired colour. This is achieved by changing the actual background colour from black to the required colour, so that when CLG clears the graphics area to the logical colour black, it actually puts a block of colour on the screen.

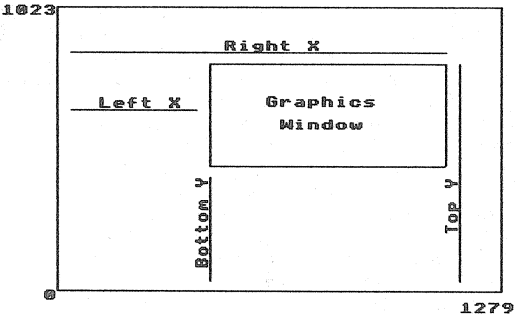
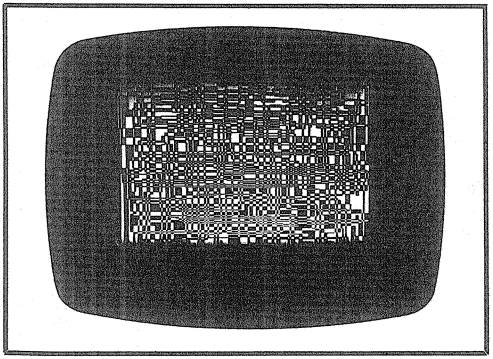


FIG 2

This technique will produce a coloured rectangle much more quickly than by using a pair of adjacent filled triangles (the commonly used technique for generating filled rectangles on the Beeb). Program one below gives a timing for the two methods. It first uses the Window Method. Line 80 defines the window, and line 90 sets the physical colour of the background to red (see User Guide pp. 262 and 222, or BEEBUG no 9 p. 3 for further details). The command CLG then clears the defined window to the chosen colour. At this point a timing is printed out, the timer is reset, and the conventional block filling method is used to draw the same sized

block. This involves drawing a pair of adjacent triangles. The first three PLOT commands on lines 150 and 160 create the first filled triangle, and the second is generated by the remaining PLOT command. This is a particularly economical way of performing the operation, since it takes advantage of the fact that in producing the first triangle, the graphics cursor actually moves to two of the points on the second triangle (since the two triangles have two points in common). The final PLOT85 call on line 160 makes use of this fact to draw the second triangle. If you run the program as it stands you will get timings of around 17 and 26 hundredths of a second respectively for the two methods, suggesting a considerable saving in time when using the window method.

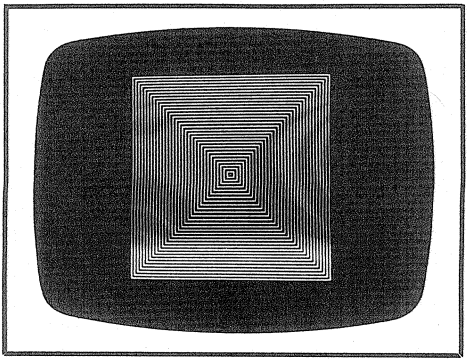
The two further programs represent brief doodlings with window-created rectangles. The first (program 2) creates random sized rectangles at random positions on the screen, while the second (program 3) produces a series of concentric squares which grow out from the centre. In both cases GCOL0,129 has been replaced by GCOL3,129. This latter achieves what is called "exclusive or" plotting, to create a more interesting effect.



```
  5 REM PROGRAM 1
 10 REM COMPARISON OF FILLED
 20 REM RECTANGLE SPEEDS
 30 MODE5
 40
 50 REM WINDOW METHOD
 60
 70 TIME=0
 80 VDU24,200;300;1000;900;
 90 GCOL0,129:CLG
100 PRINTTAB(0,25)"WINDOW TIME ";TIME
110
120 REM FILLED TRIANGLE METHOD
130
140 TIME=0
150  PLOT4,200,300:PLOT4,1000,300
160 PLOT85,200,900:PLOT85,1000,900
170 PRINTTAB(0,26)"FILLED TRIANGLE ";
TIME
180 END
```

```
  5 REM PROGRAM 2
 10 REM RANDOM BLOCK PATTERN
 20 REM USING WINDOW METHOD
 30 MODE5
 40 REPEAT
 50 REPEAT
 60 A%=RND(1280):B%=RND(1024):C%=RND(
1280):D%=RND(1024)
 70 UNTIL A%<C% AND B%<D%
 80 VDU24,A%;B%;C%;D%;
 90 GCOL3,129
100 CLG
110 UNTIL FALSE
```

```
  5 REM PROGRAM 3
 10 REM CONCENTRIC SQUARES
 20 REM USING WINDOW METHOD
 30 MODE5
 40 GCOL3,129
 50 FOR X%=0 TO 500 STEP 8
 60 VDU24,640-X%;512-X%;640+X%;512+
X%;
 70 CLG
 80 NEXT
```



CREATING ACCENTED CHARACTERS

by Alan Webster

We have received a letter from J. Austin Porter asking how to generate accented letters for foreign languages. Given below are four different ways of achieving this.

1) The first is to define your own complete accented character on an 8 by 8 grid using the VDU23 call (see 'User Defined Graphics', BEEBUG vol.1 no.2 p.9, and BEEBUGSOFT's Utilities 1 cassette which contains a character define program). For an example of this method type in this short program:

```
10 MODE 4
20 VDU 23,225,56,68,0,56,100,124,96,56
30 PRINT TAB(20,20);CHR$225
```

This creates a complete 'e' circumflex in lower case. It is printed out in line 30.

2) The second way of creating accented letters is to add an accent to existing text with a separate accent character. The way we do this is to use VDU5 to write text at the graphics cursor. The reason for this is so that we don't delete the character already there when we try to add the accent. Listed below is a program that shows the use of VDU5:

```
10 MODE 4           Selects the appropriate mode
20 VDU5             Enables us to write at the graphics cursor
30 MOVE 100,100     Moves us to a position on the screen
40 PRINT"Espana"    Prints our text
50 MOVE 224,110     Moves the graphics cursor to above the 'n' in Espana
60 PRINT"~"         Prints the accent
70 VDU4             Returns us to text mode
```

3) The third way is to combine methods 1 and 2 above by first creating just the accent, and then printing it over existing text. This is necessary, since although the BBC keyboard has a tilde, it does not have grave, circumflex and other accents.

```
10 MODE 4
20 VDU5
30 VDU 23,225,0,30,33,0,0,0,0,0
40 MOVE 100,100
50 PRINT "Espana"
60 MOVE 224,110
70 PRINT CHR$225
80 VDU4
```

This example creates an accent and stores it as character 225. The rest of the program is explained above in the second example.

4) The fourth way, which only works on operating system 1.2, is to use the TAB key to provide a complete accented character that you have defined. As an example RUN the first program and type:

```
*FX219,225    <return>
```

and then press the TAB key. This should give you the accent character. To return the TAB key to normal mode type:

```
*FX 219,9
```

In an analogous way, each of the function keys, when pressed in conjunction with the CTRL or SHIFT key can be made to produce an accented character on operating system 1.2. This is achieved by using the calls *FX225-228. See User Guide pp 439-440. If you wish to print out accented characters, then the easiest way is probably to use a printer such as the Epson FT III which has software selectable foreign character sets, or to use a daisywheel printer - many of which have accented characters on their range of daisywheels.

DISC ROUNDUP

by Colin Opie Program by Richard Russell

Disc Formatting Program
DFS Command Summary
New Error Codes

It was mentioned in last month's BEEBUG (p.2Ø) that we hoped to supply in the forthcoming issue a disc formatting program together with details of the Disc Filing System (DFS) commands. Well, here it is! Now you need not pay £3Ø for the 'privilege' of actually using the disc interface that you have already bought. Of course, if you buy Acorn's own drives then you get a manual and a formatting program. But the Acorn drives are more than a shade more expensive than those of their competitors. We are most grateful to Richard Russell of the BBC who supplied the formatting program. A formatter is an essential piece of software which must be used to format each new disc before it can be used to store programs or data. In essence this formatting operation places signals on the disc which the DFS needs in order to be able to correctly traverse a rotating disc and find the particular track(s) and sector(s) where a program or data is going to be stored and subsequently retrieved. Richard has written an efficient formatter which can be used to format 4Ø or 8Ø track discs using single or double sided drives. Even though it is written in Basic it is fast, formatting a 4Ø track disc in less than 3Ø seconds.

DISC FORMATTER

Richard has chosen to make the program readable at the expense of compactness. His intention was to make its structure understandable and easy to amend as so desired. In its present form it does not prompt for the number of tracks or the drive number. It would be very easy to incorporate such prompts, collecting appropriate values for variables 'drive' and 'tracks' (lines 7Ø and 8Ø), since Richard has used long variable names. For example the last statement on line 8Ø reads 'tracks=4Ø'. To format 8Ø track discs, just change this to 'tracks=8Ø'. To format the flip side of a double-sided disc you will need to alter the statement: drive=Ø in line 7Ø to: drive=2 (the numbering system and terminology used for accessing discs is described under DFS Command Structure below). You may wish to make this input driven by adding: 75 INPUT"Drive",drive:?par=drive . When using this modification, you could respond with Ø first time around, and then with 2.

One of the reasons for the fast operation of this formatter is that the whole disc is formatted before it is verified. Also no 'retries' are allowed if formatting or verification fails at any point. These are not unreasonable actions to take. Re-tries are rarely necessary and perhaps not even desirable as formatting is always taking place under the most favourable conditions, ie. with newly written data being immediately read on the same disc in the same drive.

The formatter uses entirely 'legal' code and should therefore work across the Tube. All direct memory accesses are to buffers allocated with the DIM statement. Now that you have a disc formatter program, all that is required to enable you to use your disc(s) is a description of the DFS commands.

THE STORY SO FAR

In previous issues of BEEBUG there have been articles on the disc system as follows:

a) DEC'82 pp.6-9, 'Disc System Review'
b) FEB'83 pp.19-2Ø, 'Disc Roundup'.

These articles show the use of '*HELP DFS' to obtain help messages from the computer, and give useful ideas on how to use the disc system efficiently. Disc hints and other related articles will continue to be a feature of the magazine. We now give a much fuller description of the DFS commands.

DFS COMMAND STRUCTURE

There are certain filename conventions used within the DFS commands which we have

to know about. With a cassette system there is only one cassette player, the side of the tape we wish to use is determined by which way we manually put the tape into the player, and so on. Disc systems give us more options. For example there could be more than one drive available, a drive might be double-sided (ie. it has two 'heads' and can therefore read both sides of a disc).

The full specification of a disc filename can be denoted by:

:n.d.name

where 'n' is the drive number ranging from 0 to 3 (and preceded by a colon), 'd' is the directory code (see below), and 'name' is the actual name of the file.

A single single-sided drive has the drive number 0. A second drive would have the drive number 1. If the drives were double-sided then the 'flip' side of the discs would be known as drive 2 and drive 3 respectively. Note therefore that the flip side of drive 0 is drive 2, and the flip side of drive 1 is drive 3.

A file may also be designated as being in a 'directory' area, the directory name being a single character. '$' (dollar) is the default directory (see later under *DRIVE and *DIR). When the filing system is started from cold drive 0 and directory '$' are assumed to be the current ones.

File names can be up to 7 alphanumeric characters in length - which is extremely short - and you cannot use the special characters hash (#), asterisk (*), full stop (.) or colon (:). The reason why the full stop and colon cannot be used is because, as is seen from the filename specification above, they are used as delimiters. A hash sign can be used in certain commands to take the place of any character for matching purposes, and an asterisk can be used for the same purpose in place of any sequence of characters, (up to a delimiter). These last two special symbols are known as 'wildcards'.

Note that the drive number and/or the directory letter, together with their delimiters, can be left off a filename specification provided you are willing to let the system use the assumed values. For example if the current drive is drive 2 and the current directory is 'G' then the following commands will be identical in effect:

```
SAVE":2.G.NAME"        SAVE":2.NAME"
SAVE"G.NAME"           SAVE"NAME"
```

This feature may take a little getting used to at first, but it soon becomes fairly natural. The options available can be put to very good use once they have been mastered.

DISC FILING SYSTEM COMMANDS

The official User Guide on the DFS runs to 86 pages and therefore it is clearly not possible to give as much detail here. On the other hand we give below a nucleus of information which, once known, will allow you to use a very large percentage of the facilities offered by the DFS.

Given below is a list of the commands available, their minimum abbreviation (the full stop must be included), and their purpose. Arguments (parameters which must be entered after the main command) must be separated by a space, except where otherwise stated. The term 'fspec' refers to a legal file specification (as discussed above). WARNING: all commands with the symbol '[+]' against the text below can result in the destruction of programs or data currently held in memory.

*ACCESS	*A.	Locks or unlocks a file. If a file is locked then it cannot be written over. It is locked by using the form: *ACCESS fspec L and unlocked by omitting the 'L' in the command.
*BACKUP	*BAC.	Will enable the entire contents of one disc to be backed up (ie. copied) on another disc. Two arguments are required,

		the source and destination drive numbers. See also *ENABLE. [+]
*BUILD	*BU.	The argument to BUILD is 'fspec'. All subsequent 'auto' entered text will be placed into the specified file.
*CAT	*.	Optional drive number as an argument. It supplies a catalogue of the specified (or current) drive.
*COMPACT	*COM.	Optional a drive number as an argument. This takes all the data and programs on the specified disc and closes them up together. In this way there is no wasted space on the disc. The files can still be accessed individually afterwards. This is the command to use if you find that your disc is 'full' - it probably isn't, it just requires the stored information to be compacted (but see warning about *COMPACT in BEEBUG no.9, p.2Ø). [+]
*COPY	*COP.	Three arguments are required for this command. The first is the source drive number, the second is the destination drive number, and the third is 'fspec'. This is one command where the wildcard characters (hash and dollar) become very useful in the copying of multiple files. Eg: *COPY Ø 1 DATA2 will copy the file 'DATA2' from drive Ø to drive 1. [+]
*DELETE	*DE.	This will remove the specified single file from the disc area. The argument is therefore 'fspec' and wildcards may not be used.
*DESTROY	*DES.	This has the same form of argument as *DELETE (ie, 'fspec') with the exception that wildcards may be used, so as to delete a possible multiple number of files. See also *ENABLE.
*DIR	*DI.	Requires a directory character as its argument, and will set the 'current' directory to this value.
*DRIVE	*DR.	The argument is a valid drive number. The current 'assumed' drive will be set accordingly.
*DUMP	*DU.	The argument is 'fspec' and it will produce a hexadecimal listing (dump) of the specified file on to the screen. It is useful to set 'page' mode before calling this command. See also the commands *LIST and *TYPE.
*ENABLE	*EN.	Two commands, *BACKUP and *DESTROY, cannot be used unless they are previously 'enabled'. This command performs the enabling. No arguments are required.
*EXEC	*E.	A text file may be held on disc and then read as though the text were coming from the keyboard. A convenient way of getting the text on to disc is through use of the *BUILD command. This facility is a convenient one to use for merging programs that have been spooled (see BEEBUG no.9, p.2Ø). The argument to *EXEC is 'fspec'.
*HELP	*H.	This requires an argument consisting of either or both of the keywords DFS and UTILS. It displays useful information about the commands available in the disc system and is therefore a good 'quick reference' guide.
*INFO	*I.	The argument is 'fspec' and the command displays information about the specified file(s). More information is given than by using *CAT and it comes in the following order: Directory, Filename, Access code, Load address, Execution address, Length (bytes), and Start sector.
*LIB	*LIB	A machine code program on disc may be loaded and run in one complete operation merely by typing *name (where 'name' is the name of the program), provided it exists on the drive and in the directory given by the 'library' area specification of the current drive. For example, if a *CAT on the current drive (say, drive Ø) shows the library area specification to be :Ø.X then a program with the file

		specification :Ø.X.MERGE can be loaded and run by merely typing *MERGE. The command *LIB accepts the form - :n.d , where 'n' and 'd' are the drive number and directory which is to be used as the future library area.
*LIST	*LIST	This command is related to *TYPE and *DUMP in that they are all 'listing' commands. *LIST will display on the screen the specified text file, with line numbers. The argument is therefore 'fspec'.
*LOAD	*L.	This command is the disc equivalent of the same command within the Cassette system and has the same form, except that the cassette filename must be replaced by a valid 'fspec'.
*OPT 1	*O.1	It is possible to get the system to display the information normally given by *INFO whenever a file is accessed. *OPT 1 1 will enable this feature, and *OPT 1 Ø will disable it.
*OPT 4	*O.4	This command will select the auto-start option for the current drive. Four options are available: *OPT 4 Ø does nothing (off) *OPT 4 1 will *LOAD !BOOT *OPT 4 2 will *RUN !BOOT *OPT 4 3 will *EXEC !BOOT Note that the file !BOOT is therefore a special file as far as the system is concerned (see Auto Start article in BEEBUG no.9, p.19).
*RENAME	*RE.	Enables the name of a file to be changed and/or its directory (but not its drive). Two arguments are required, each one being a 'fspec'. The first is the original name and the second is the new name required. Note that the drive number would normally be omitted from the file specifications, but if included they must refer to the same drive.
*RUN	*R.	This command is the disc equivalent of the same command within the Cassette system and has the same form, except that the cassette filename must be replaced by a valid 'fspec'.
*SAVE	*S.	This command is the disc equivalent of the same command within the Cassette system and has the same form, except that the cassette filename must be replaced by a valid 'fspec'.
*SPOOL	*SP.	This has two forms, one with an argument ('fspec') and the other without. *SPOOL with an argument will open up the specified file to accept all characters sent to the screen (ie. it is like a printer echo except that o/p is going to a disc text file). The file is closed by executing *SPOOL without an argument. One use for this command is to obtain a text copy on disc of a Basic program. Another use is in the merging of Basic programs (see BEEBUG no.9, p.2Ø).
*TITLE	*TI.	Each disc may have a title of up to 12 characters. This command accepts a string as an argument and uses the first 12 characters for entitling the disc. Quotes are only necessary around the string if it contains spaces. You can obtain coloured titles - see disc hint in this issue.
*TYPE	*TY.	This command is the same as *LIST except that the text file is listed without line numbers. For a useful application of this command see BEEBUG no.9, p.19.
*WIPE	*W.	This is similar to *DESTROY except that you get the opportunity to confirm the deletion of any file which matches the specification. Type a 'Y' if you still want the file erased or 'N' if not. Note that the abbreviated form will not work if you have Wordwise in your machine.

Note that the above '*' commands are pure Filing System commands and are in addition to the standard Basic commands LOAD, SAVE, CHAIN, BGET etc., which will work with both Cassette and Disc systems. Once the DFS is fitted you can type the command *TAPE to get the machine to think that it is a Cassette system and then *DISC to return to a disc system. This provides a very easy mechanism for transferring files from cassette to disc and vice versa.

DISC FAULT ERROR MESSAGES

There are a number of error messages generated by the DFS, of which most are self explanatory. Two of these are 'umbrella' type error messages which merely give a vague reason for the error followed by the relevant disc interface error code. They have the form:

Disc fault NN at TT/SS
Drive fault NN at TT/SS

where NN is the error code and TT/SS is the track and sector where the error was found. The following table, supplied by Richard Russell, gives a list of the NN error codes (in hexadecimal) together with their meaning and probable cause:

CODE	MEANING	PROBABLE CAUSE
08	Clock error	Damaged disk / Faulty disk drive
0A	Late DMA	Faulty computer
0C	ID CRC error	Damaged disk / Faulty disk drive
0E	Data CRC error	Damaged disk / Faulty disk drive
10	Drive not ready	Faulty disk drive
12	Write protect	Disk write protected
14	Track 0 not found	Faulty disk drive
16	Write fault	Faulty computer
18	Sector not found	Disk not formatted or incorrectly formatted

The codes have been deduced from the data book on the disc interface controller IC, and it is not clear whether all of them can be produced by the DFS. This is particularly true of code 12 (Hex) which seems to be intercepted by the DFS in order to produce the more 'friendly' message of 'Disc read only'.

FINALLY

As mentioned above, BEEBUG will be carrying a regular feature on disc use (and abuse!). The above information should be enough to get you started, and then perhaps you can start to share your own hints and tips.

```
  10REM. Program to format a disk
  20REM. (alternative to FORM40)
  30REM. by R.T.Russell, BBC, 17-02
-1983.
  40:
  50DIM par 15,buffer 255:X%=par MO
D 256:Y%=par DIV 256:A%=&7F
  60:
  70MODE 7:WIDTH 0:drive=0:osword=&
FFF1:?par=drive
  80head=0:gap3=27:gap1=22:length=1
:sectors=10:tracks=40
  90:
 100PRINT '"This program formats t
he disk in drive ";drive
 110PRINT "Do you really want to fo
rmat the disk? "
 120ON INSTR("YyNn",GET$) GOTO 130,
130,890,890 ELSE 120
 130:
 140PRINT '"Formatting drive ";driv
e'
 150FOR track=0 TO tracks-1
 160@%=4:PRINT track;:@%=&90A
 170PROCseek(track)
 180PROCformat(track)
 190NEXT track
 200:
 210PRINT '"Verifying drive ";drive

 220FOR track=0 TO tracks-1
 230@%=4:PRINT track;:@%=&90A
 240PROCverify(track)
 250NEXT track
 260:
 270PRINT '"Initialising drive ";dr
ive
 280PROCinitialise
 290PRINT '"Finished."
 300END
 310:
 320DEF PROCseek(track)
 330par!1=buffer
 340par?5=1
 350par?6=&69
 360par?7=track
```

```
  370CALL osword
  380?par=-1
  390ENDPROC
  400:
  410DEF PROCformat(track)
  420FOR sector=0 TO sectors-1
  430ident=buffer+4*sector
  440ident?0=track
  450ident?1=head
  460ident?2=sector
  470ident?3=length
  480NEXT sector
  490par!1=buffer
  500par?5=5
  510par?6=&63
  520par?7=track
  530par?8=gap3-6
  540par?9=sectors+32*length
  550par?10=0
  560par?11=gap1-6
  570CALL osword
  580IF par?12 PRINT'"Formatting err
or"':END
  590ENDPROC
  600:
  610DEF PROCverify(track)
  620par!1=buffer
  630par?5=3
  640par?6=&5F
  650par?7=track
  660par?8=0
  670par?9=sectors+32*length
  680CALL osword
  690IF par?10 PRINT'"Verify error"'
:END
  700ENDPROC
  710:
  720DEF PROCinitialise
  730FOR i=0 TO 252 STEP 4
  740buffer!i=0
  750NEXT i
  760par!1=buffer
  770par?5=3
  780par?6=&4B
  790par?7=0
  800par?8=0
  810par?9=1+32*length
  820CALL osword
  830buffer?7=sectors*tracks
  840buffer?6=(sectors*tracks)DIV 256
  850par?8=1
  860IF par?10=0 CALL osword
  870IF par?10 PRINT'"Initialise err
or"':END
  880ENDPROC
  890END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

NOT TRUE?

The User Guide on page 100 declares that all non-zero values are regarded as TRUE. John Dove has written to point out that this is not exactly correct for non-integers. A conversion to integer form takes place before the test, and thus any number between -1 and +1 (but not including) is recognised as FALSE, eg:

```
10 H=0.5
20 IF H THEN PRINT"YES" ELSE PRINT"NO"
RUN
NO
```

DEBUGGING WITH EVAL

The following routine from G.Weston of Clwyd is useful if your program needs debugging. When called by the main program it asks for the variable name that you wish to test and prints out the contents of it.

```
2000 DEF PROCTEST
2010 INPUT"Var."Z$
2020 IF Z$<>"999" PRINT EVAL(Z$):GOTO 2010
2030 ENDPROC
```

To use this procedure just insert the line PROCTEST at the trouble spot and you can then test the variables.

NON-ACCESSIBLE FILENAMES

When using Wordwise if you select option 1 to save the file on disc, and then accidently enter *. to obtain a catalogue, you will end up with a file on disc apparently called #.* try as you may you cannot delete this in the ordinary way, so you are stuck with it. The following run will delete it from the disc.

```
DIM X% 30: Y%=X% DIV 256: $X%="DELETE *.":CALL &FFF7
```

The same problem could arise with other packages currently on the market, so be careful.

SW

Program tested on O.S. 0.1 and 1.2

PASSING FILENAMES TO THE MOS

by John Yale

John Yale explains a way of using variable filenames from within a program. This is an essential requirement of any data handling program but cannot be directly performed on the BBC micro.

When using the *SAVE and *LOAD commands from within a Basic program it is not possible to use a Basic string as a filename, or a Basic variable as an address. The reason for this is that the entire line after the '*' is passed to the MOS which knows nothing about Basic variables.

However the MOS contains a routine OSCLI, described on page 463 of the manual, which allows any MOS command string to be passed to it by address. Therefore if we build the command string in memory before passing it to OSCLI then any Basic strings or variables may be used as file names.

The demonstation program below shows this being done for *LOAD and *SAVE (Note that the * is not needed in the command line) with filename F$. The program saves and loads the function key buffer at &0B00.

To use this technique, include lines 40,50 and the procedure PROCoscli in your program, and call as shown in lines 140 and 190. To understand the multiple quotes, remember that double quotes ("") within a quoted string effectively places one set of quotes (") in that string. So """" is a string containing just " .

To use Basic variables as the address in the command line, use may be made of STR$. It is not generally known that ~ may be used with STR$ to force conversion in hexadecimal as required by the *SAVE and *LOAD commands. eg:

```
A%=100
A$=STR$(A%)
B$=STR$~(A%)
PRINT A$,B$
```

A call to save a file F$ from address S% to E% would be:

```
PROCoscli("SAVE"""+F$+""""+STR$~(S%)+" "+STR$~(E%))
```

To see how this command line is made up try typing:

```
F$ = "TEST"
S% = &1234
E% = &5678
PRINT"SAVE"""+F$+""""+STR$~(S%)+" "+STR$~(E%)
```

when you will see printed: SAVE"TEST"1234 5678

which is the correct command. It is advisable to check programs of this type in this way before running them, copying the relevant code into the PRINT statement directly from the program with the COPY key.

The technique described above can be used to insert a string variable in any * command; and another use of the procedure PROCoscli is to set a function key with a string containing the contents of a Basic string variable. eg:

```
PROCoscli("KEY"+STR$(N%)+K$)
```

programs key N with string K$.

Note that if your machine is fitted with Basic version 2, there is a more direct way of addressing the command line interpreter. The same principles apply, but see BEEBUG no.6 p.11.

```
 10 REM Passing filenames to the MOS
 20
 30 REM Allocate buffer for command line
 40 oscli%= &FFF7
 50 DIM buf% 30
 60
 70 INPUT"LOAD or SAVE",A$
 80 IF A$="LOAD" THEN PROCload
 90 IF A$="SAVE" THEN PROCsave
100 GOTO 70
110 DEFPROCsave
120 INPUT"Filename to save ",F$
130
140 PROCoscli("SAVE"""+F$+""" 0B00 0BFF")
150 ENDPROC
160 DEFPROCload
170 INPUT"Filename to load ",F$
180
190 PROCoscli("LOAD"""+F$+""""")
200 ENDPROC
210
220 END
230 DEFPROCoscli(cline$)
240 $buf% = cline$
250 X%=buf%
260 Y%=buf% DIV 256
270 CALL oscli%
280 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CONTROL CHARACTERS

If you press the CTRL key at the same time as certain keyboard letters, you can generate some useful control codes, enabling you for example to clear the screen or home the cursor to the top left hand corner, all at a stroke. Here are some of the more useful but less well known ones:

CTRL	G	Beep (same as VDU7)
CTRL	H	Cursor left (non-erasing)
CTRL	I	Cursor right
CTRL	J	Cursor down
CTRL	K	Cursor up (once it reaches the top it will scroll the screen down)
CTRL	L	Clear text screen (perhaps the most useful)
CTRL	M	Return
CTRL	P	Clear graphics screen
CTRL	V	Followed by a number 0 to 7 will execute a mode change - but use it advisedly since it does not reset HIMEM - giving the wrong memory allocation for the chosen mode.

Note that all of these control characters may be embedded in the user keys by use of the | character. For example |M is 'return' |G is 'beep'. See also the User Guide pp.488-490.

MUSICAL DATA STATEMENTS

When creating music on the Beeb it is very wasteful to keep on typing in SOUND statements. A much better method is to use DATA statements to hold the information about the notes. An example of this was supplied by Tony Fryer (aged 12), which gives a never ending rendering of the popular song 'Lazy-bones':

```
 10 REPEAT:REPEAT
 20 READ P,D
 30 SOUND 1,-15,P,D
 40 UNTIL P<0
 50 RESTORE
 60 UNTIL FALSE
100 DATA 157,12,145,5,129,20,145,7
110 DATA 149,3,153,7,157,3,129,20
120 DATA 145,7,149,3,153,7,157,3
130 DATA 149,10,141,10,129,10,177,10
140 DATA 157,20,145,7,149,3,145,7
150 DATA 149,3,145,10,149,10,137,40
160 DATA 137,0,137,7,145,3,137,7
170 DATA 145,3,137,10,129,10,129,0
180 DATA 129,20,-1,0
```

Program tested on O.S. 0·1 and 1·2

ARTILLERY DUEL (16k/32k)

by Colin Walton

This is a simple 2-D game which sets the scene for a duel between two artillery barrages, each barrage being shown as a single turret station. It is really a game for two players. Each player becomes the gunnery commander for the appropriate barrage. Each commander has alternate opportunities at trying to estimate:

a) the necessary angle of his gun turret, and

b) the number of powder bags required to hit his opponents gun.

Each entry is terminated by pressing <return>. For each duel there is an invariable wind direction and speed, displayed at the top of the screen, which must be taken account of in your estimations as gunnery commander. Previous estimations are left on the screen so that a clear record is kept of past attempts.

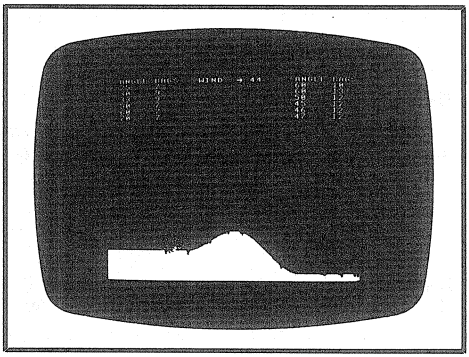
A nice feature of the game is the way in which it randomly creates different landscapes for each duel. Some of the resulting landscapes require considerable skill in gunnery, especially in view of the variety of possible wind directions and strengths.

If you have a 32K machine you can obtain a greater range of colours by setting Mode 1 in line 60.

LIST OF MAJOR PROCEDURES

PROCEDURE	USE
Ld	Draws the random landscape by using function 'Sfc'
Cs	Sets the screen colours and gives the headings
Gps	Draws the gun positions
Fr	This accepts the values for a shot and then fires the missile, checking for a hit
HIT	Called by procedure 'Fr', and displays an exploding gun.

```
  20 MODE7:VDU23;11,0;0;0;0
  30 PRINTTAB(10,10)CHR$141"ARTILLERY DUEL"'TAB(10)CHR$
141"ARTILLERY DUEL"TAB(10)CHR$141"                "
  40 TIME=0:REPEAT:UNTILTIME=200
  50 CLS
  60 MODE4:VDU23;11,0;0;0;0
  70 VDU19,0,4,0,0,0,19,3,2,0,0,0
  80 RY%=1:HIT%=0
  90 WE%=RND(400):EE%=RND(400):WH%=4*RND(160)-4:EH%=1280
-4*RND(160):WS%=RND(50)
 100 HE%=(RND(640-(EE%+WE%)))/2
 110 WG%=RND(640)-1:WGY%=FNSfc(WG%)
 120 EG%=1241-RND(601):EGY%=FNSfc(EG%)
 130 IFRND(1)>0.5THENWS%=-WS%
 140 IFRND(1)>0.5THENR%=1ELSER%=-1
 150 GCOL0,3
 160 PROCLd
 170 PROCCs
 180 PROCGps
 190 REPEATIFR%=1PROCFr(EG%,EGY%,WG%,WGY%)ELSEPROCFr(WG%
,WGY%,EG%,EGY%)
 200 UNTILHIT%=1
 210 GOTO60
 220 END
 230 DEFPROCLd
 240 FORXX%=0TO1280STEP4
 250 MOVEXX%,0
 260 DRAWXX%,FNSfc(XX%)
 270 NEXT
 280 ENDPROC
 290 DEFFNSfc(XX%)
 300 IFXX%<WH%THEN=WE%
 310 IFXX%>=EH%ANDXX%<=1279THEN=EE%
```



```
 320 W%=EH%-WH%
 330 I%=WE%+HE%
 340 J%=-HE%*COS(RAD((XX%-WH%)*360/W%))
 350 K%=(XX%-WH%)*(EE%-WE%)/W%
 360 =I%+J%+K%
 370 DEFPROCGps
 380 GCOL0,1
 390 VDU5:MOVEWG%,FNSfc(WG%)+30:PRINTCHR$224:MOVEEG%-20,
FNSfc(EG%-20)+30:PRINTCHR$225:VDU4
 400 ENDPROC
 410 DEFPROCCs
 420 VDU23,224,1,2,4,8,24,56,120,255
 430 VDU23,225,128,64,32,16,24,28,30,255
```

```
 440 VDU23,226,24,12,6,255,255,6,12,24
 450 VDU23,227,24,48,96,255,255,96,48,24
 460 COLOUR1
 470 PRINT"  ANGLE BAGS   WIND    ";ABS(WS%);TAB(29)" A
NGLE BAGS"
 480 IFWS%>=0THENPRINTTAB(21,0)CHR$226ELSEPRINTTAB(21,0)
CHR$227
 490 COLOUR1
 500 PRINTTAB(0,0)CHR$224TAB(28,0)CHR$225
 510 ENDPROC
 520 DEFPROCFr(GUN%,GUNY%,TAR%,TARY%)
 530 COLOUR1
 540 VDU23;11,255;0;0;0
 550 VDU31,15+14*R%,RY%
 560 INPUT" "ANGLE
 570 VDU31,21+14*R%,RY%
 580 INPUT" "BAGS
 590 VDU23;11,0;0;0;0
 600 TIME=0:GCOL0,1
 610 XA%=GUN%:YA%=GUNY%
 620 TIM=TIME/10
 630 ZR%=TIM*TIM/2000*WS%
 640 XR%=-R%*BAGS*TIM*COS(RAD(ANGLE))+ZR%
 650 YR%=BAGS*TIM*SIN(RAD(ANGLE))-0.5*TIM*TIM/10
 660 X%=GUN%+XR%:Y%=GUNY%+YR%
 670 MOVEX%+4,Y%-4:MOVEX%-4,Y%-4:PLOT85,X%+4,Y%+4:PLOT85,
X%-4,Y%+4
 680 MOVEXA%+4,YA%-4:MOVEXA%-4,YA%-4:PLOT87,XA%+4,YA%+4
:PLOT87,XA%-4,YA%+4
 690 PROCGps
 700 XA%=X%:YA%=Y%
 710 IFX%>TAR%-10ANDX%<TAR%+10ANDY%>TARY%-10ANDY%<TARY%
+10THENGCOL0,3:PROCHIT:ENDPROC
 720 IFY%<FNSfc(X%)THENGOTO740
 730 IFX%>0ANDX%<1280THENGOTO620
 740 SOUND0,-15,6,2
 750 GCOL0,0
 760 MOVEX%,FNSfc(X%)-10:MOVEX%-5,FNSfc(X%-5):PLOT85,X%
+5,FNSfc(X%+5)
 770 IFR%=1THENRY%=RY%+1
 780 R%=-R%
 790 ENDPROC
 800 DEFPROCHIT
 810 HIT%=1
 820 ENVELOPE1,2,0,0,0,0,0,0,127,0,0,-1,0,126:SOUND0,1,
4,5:SOUND0,1,5,10:SOUND0,1,6,15
 830 FORC%=0TO150
 840 DIST%=RND(100):ANG=RAD(RND(360))
 850 PLOT69,X%+DIST%*COS(ANG),Y%+DIST%*SIN(ANG)
 860 NEXT
 870 TIME=0:REPEATUNTILTIME=500
 880 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

FUNCTION KEY LABELS

As users will know, it is possible to place a length of paper under the plastic strip above the function keys, with notes written on the paper indicating the use of each function key. However, this can be rather too much bother if you are in the habit of changing the definitions frequently. An answer to this problem has been contributed by Mr. B A Etherington. He suggests writing directly on the plastic strip with a water soluble felt pen. When you change the definition simply wipe off the old writing with a damp cloth and write on the new definition. A piece of blank white paper underneath the plastic strip makes the writing much more easily visible. [Be careful not to use ink which cannot easily be wiped off. Ed.]

MORE LIVES IN ROCKET RAID

Heres a tip for those people who need lots of practice at cracking the final sector in Acornsoft's "Rocket Raid". Do the following:

```
PAGE=&4000
LOAD"RAID2"
1850 ?&1178=&7F
RUN
```

This will give you 128 lives! - John Harris.
[We cannot verify the above because we don't yet have a copy of Rocket Raid. Ed.]

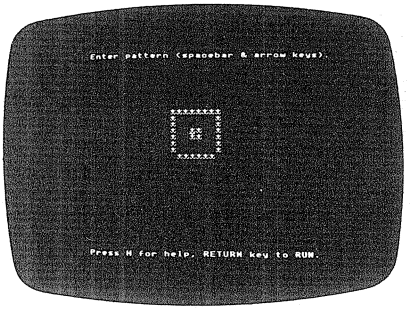
MAKE-SHIFT DOLLAR SIGNS

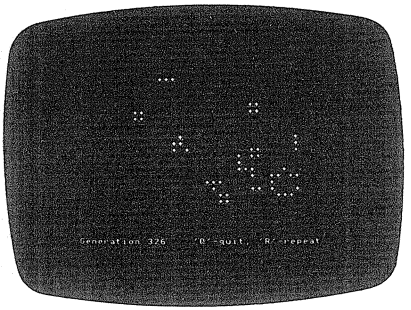
To anyone who has either a typewriter or a printer without a dollar sign, you may like to try the sequence of a capital 'S', a backspace, then a slash sign '/'. Although it doesn't look exactly the same as a dollar sign, it gives a fair representation.

Program tested on O.S. 0.1 and 1.2

LIFE (32k)

Text by Sheridan Williams, Program by Jack Hardie





Life is a game that was developed by Professor John Conway at the University of Cambridge. As far as I am aware it was first published in the "Mathematical Games" feature in "Scientific American" magazine by Martin Gardner in October 1970.

The game is played on a grid, where each cell may be alive or dead. Each cell has 8 neighbours. The population of the cells change by a set of predetermined rules which are as follows:

1. If a live cell has 2 or 3 live neighbours it will live on to the next generation.
2. If an empty (dead) cell has exactly 3 neighbours the cell will be born in the next generation.
3. If a cell has none or 1 neighbour it will die of loneliness.
4. If a cell has 4 or more neighbours it will die from overcrowding.

The study of 'LIFE' has become a cult and there are numerous references to it in all sorts of magazines. Here are some references for further reading:
BYTE magazine Sept 75, Oct 75, Dec 75, Jan 76, Dec 78, Jan 79;
SCIENTIFIC AMERICAN Oct 70, Nov 70, Jan 71, Feb 71, Mar 71, Apr 71, Nov 71, Jan 82.
There must be other references too in British magazines.

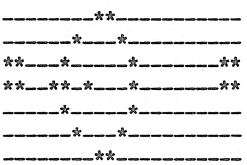
There is even a set of jargon built up around LIFE, for example various shapes can be divided into classes like Constellation, Still Life, Oscillator, Methuselah, Spaceship, Uniform Propagators, Eaters etc.

Here are two examples, the first is of a Spaceship called a 'glider', the second is one of the record holders for a long lived "Methuselah" system:

```
GLIDER      METHUSELAH
 ***         -*-----
 --*         ---*---
 -*-         **--***
```

The glider will move slowly across the screen. The Methuselah will just expand, for ever?

There are also 'puffer trains', 'glider guns' and many others. Try and invent some of your own. As a last example here is a basic shuttle. The four blobs at either end are 'eaters' and are used to clear up the debris. The central shape will change, but it will move regularly between the left and right 'eaters' for ever.



We will give £20 for the first pattern that produces the most interesting (in the editors' view) pattern in less than 100 generations using this version of LIFE. Entries must be on a postcard together with your name and address - no replies can be given, but we will publish a selection of the best: send to LIFE competition, PO Box 50, St Albans, Herts.

The particular implementation of LIFE listed below combines Basic and assembler in an intelligent way. The most time critical operations are in assembler, and the other parts are in Basic for ease of writing. The program also makes good use of procedures. Note how the three main assembler routines are used with the statements: CALL zero, CALL breed and CALL screen in lines 260,330 and 350 respectively.

Be very careful when typing in lines 900-2520; and make sure that you save a copy before actually running the program because an error in the assembler part can disable the system.

```
  10 REM LIFE in BASIC and assembler.
  20 REM Jack Hardie 11 October 1982.
  30 REM Life pattern is stored in a
  40 REM buffer extending down from
  50 REM location 'topleft' (which
  60 REM corresponds to top left
  70 REM character of screen) for
  80 REM 31*40 memory locations
  90 REM (corresponding to 31 rows
 100 REM on the screen). Last screen
 110 REM row is left blank for text.
 120 REM Live positions on screen are
 130 REM shown by frog-like character.
 140 REM In memory, live is 1, dead is
 150 REM 0. Bit 0 is current generation
 160 REM bit 1 is the next.
 170
 180 MODE 4
 190 HIMEM=HIMEM-&500
 200 DIM note(8)
 210 PROCASSEMBLE
 220 PROCVDU
 230 PROCLOGO
 240 REPEAT
 245 PROCVDU
 250 gen=-1
 260 CALL zero
 270 PROCPATTERN
 280 VDU 23;8202;0;0;0;:REM cursor off
 290 CLS
 300 PRINT TAB(19,31);"'Q'-quit, 'R'-repeat";TAB(1,31)
"Generation ";
 310 REPEAT
 320 gen=gen+1
 330 CALL breed
 340 PRINT TAB(12,31);gen;
 350 CALL screen
 360 key=INKEY(0) AND &DF
 370 UNTIL key=&51 OR key=&52
 380 VDU22,4:REM Enable cursor
 390 UNTIL key=&51
 400 MODE 7
 410 PRINT TAB(15,10);"Thats all."
 420 END
 430
 440 DEF PROCVDU
 450 VDU19,0,4,0,0,0:REM Blue ground
 460 VDU19,1,7,0,0,0:REM White f'ground
 470 VDU 23,224,8,&5D,&3E,&1C,&49,&55,&22,0:REM frog ch
aracter
 480 ENDPROC
 490
 500 DEF PROCPATTERN:REM Enter pattern.
 510  LOCAL key
 520 REM Disable editing
 530 *FX4,1
 540 CLS:PROCPROMPTS
 550 REPEAT
 560 key=GET
 570 IF key<&8C AND key>&87 THEN PROCMOVE(key)
 580 IF key=&7F THEN PROCUNFROG
 590 IF key=&20 THEN PROCFROG
 600 IF (key AND &DF)=&48 THEN PROCHELP:CALL zero:PROCP
ROMPTS
 610 UNTIL key=&0D
 620 REM Enable editing.
 630 *FX4,0
 640 ENDPROC
 650
 660 DEF PROCMOVE(k)
 670  VDU(k AND &F)
 680 ENDPROC
 690
 700 DEF PROCFROG
 710 ?(topleft-POS-40*VPOS)=1
 720 VDU 224,8
 730 ENDPROC
 740
 750 DEF PROCUNFROG
 760 VDU 9,&7F
 770 ?(topleft-POS-40*VPOS)=0
 780 ENDPROC
 790
 800 DEF PROCASSEMBLE
 810 oswrch=&FFEE
 820 pointer=&80
 830 count=&82
 840 topleft=HIMEM+&500-40
 850 toplo=topleft AND &FF
 860 tophi=(topleft AND &FF00)/&FF
 870 DIM Q% 250
 880 FOR pass=0 TO 2 STEP 2
 890 P%=Q%
 900 [OPT pass
```

```
 910 .screen
 920 \ Print LIFE pattern on screen
 930 \ and shift each memory loc'n
 940 \ right so that next generation
 950 \ becomes bit zero.
 960
 970 \ Memory is read downwards from
 980 \ 'topleft' down to first page
 990 \ boundary, then in 4 blocks of
1000 \ one page each.
1010
1020 LDA #30          \ Cursor home
1030 JSR oswrch
1040 LDA #0
1050 STA pointer
1060 LDA #tophi
1070 STA pointer+1
1080 LDX #5           \ Block number
1090 LDY #toplo
1100 .nextchar
1110 LDA (pointer),Y
1120 LSR A            \  Next gen-
1130 STA (pointer),Y  \  -eration
1140 JSR writechar
1150 DEY
1160 BNE nextchar
1170 LDA (pointer),Y  \ Y=0 is a
1180 LSR A            \ special case
1190 STA(pointer),Y
1200 JSR writechar
1210 DEY
1220 .nextblock
1230 DEC pointer+1    \ Block ptr.
1240 DEX              \ Block count.
1250 BNE nextchar
1260 LDA #30          \ Cursor home.
1270 JSR oswrch
1280 RTS
1290
1300 .writechar
1310 BCS frog
1320 LDA #&20         \ ASCII space.
1330 JSR oswrch
1340 RTS
1350 .frog
1360 LDA #224         \ ASCII frog.
1370 JSR oswrch
1380 RTS
1390
1400 .breed
1410 \ Apply LIFE rules, starting
1420 \ at screen row 3, column 2.
1430 \ Each of 8 neighbours is
1440 \ located by an offset from
1450 \ south-east neighbour.
1460
1470 CLD
1480 LDA #toplo-82    \ Start loc'n.
1490 STA pointer
1500 LDA #tophi
1510 STA pointer+1
1520 LDX #5           \ 5 blocks.
1530 .nextcell
1540 JSR counter      \ Count n'bours
1550 JSR check        \ Alive ?
1560 DEC pointer
1570 BNE nextcell
1580 JSR counter      \ Zero case.
1590 JSR check
1600 DEC pointer
1610 DEC pointer+1    \ Next block.
1620 DEX
1630 BNE nextcell
1640 RTS
1650
1660 .counter
1670 CLC
1680 LDY #82          \ Offset for
1690                  \ N-West cell.
1700 LDA #1
1710 AND (pointer),Y
1720 STA count
1730 DEY              \ North cell.
1740 LDA #1
1750 AND (pointer),Y
1760 ADC count
1770 STA count
1780 DEY              \ N-East cell.
1790 LDA #1
1800 AND (pointer),Y
1810 ADC count
1820 STA count
1830 LDY #42          \ West cell
1840 LDA #1
1850 AND (pointer),Y
1860 ADC count
1870 STA count
1880 DEY:DEY          \ East cell.
1890 LDA #1
1900 AND (pointer),Y
1910 ADC count
1920 STA count
1930 LDY #2           \ S-West cell.
1940 LDA #1
1950 AND (pointer),Y
1960 ADC count
1970 STA count
1980 DEY              \ South cell.
1990 LDA #1
2000 AND (pointer),Y
2010 ADC count
2020 STA count
2030 DEY              \ S-East cell.
2040 LDA #1
2050 AND (pointer),Y
2060 ADC count
2070 STA count
2080 RTS
2090
2100 .check           \ Alive or dead?
2110 LDY #41          \ Offset for
2120                  \ centre cell.
2130 LDA #3
2140 CMP count
2150 BEQ live         \ 3 neighbours.
2160 LDA #2
2170 CMP count
2180 BEQ same         \ 2 neighbours.
2190 .dead            \ NOT 2 OR 3.
2200 LDA #1
2210 AND (pointer),Y  \ Set bit 0.
2220 STA (pointer),Y
2230 RTS
2240 .same LDA #1
2250 AND (pointer),Y  \ Test bit 0.
2260 BEQ dead
2270 .live
2280 LDA #2
2290 ORA (pointer),Y  \ Reset bit 1.
2300 STA (pointer),Y
2310 RTS
2320
2330 .zero
2340 \ Routine to set memory buffer
2350 \ to zero.
2360 LDA #tophi
2370 STA pointer+1
2380 LDA #0
2390 STA pointer
2400 LDY #0
2410 LDX #5
2420 .nextzero
2430 STA (pointer),Y
2440 DEY
2450 BNE nextzero
2460 STA (pointer),Y
2470 DEY
```

```
2480 DEC pointer+1
2490 DEX
2500 BNE nextzero
2510 RTS
2520 ]
2530 NEXT pass
2540 ENDPROC
2550
2560 DEF PROCHELP
2570 LOCAL key
2580 REM Disable auto key repeat.
2590 *FX11,0
2600 CLS
2610 PRINT"                    LIFE"
2620 PRINT"                    ----"
2630 PRINT
2640 PRINT"  The world of LIFE is populated with "
2650 PRINT"small frog-like creatures which only"
2660 PRINT"show up on the V.D.U screen if they are"
2670 PRINT"'alive'. Each position on the screen"
2680 PRINT"lives or dies according to how many "
2690 PRINT"live squares are next to it, so if you"
2700 PRINT"start with a pattern of live creatures"
2710 PRINT"the pattern will (usually) change after"
2720 PRINT"each 'generation' and the pattern of"
2730 PRINT"life will 'evolve'."
2740 PRINT
2750 PRINT"Here are the rules of life, they apply"
2760 PRINT"to each character position on the"
2770 PRINT"screen:"
2780 PRINT
2790 PRINT"  NUMBER"
2800 PRINT"    of",,"       RESULT       "
2810 PRINT"NEIGHBOURS"
2820 PRINT"-------------------------------------"
2830 PRINT
2840 PRINT"  0 or 1      Death from loneliness"
2850 PRINT"     2        No change"
2860 PRINT"     3        Breed a new frog"
2870 PRINT"  4 to 8      Death from overcrowding"
2880 PRINT
2890 PRINT" Press spacebar to continue"
2900 PROCWAIT(200)
2910 REM Flush input buffer
2920 *FX15,1
2930 key=GET
2940 CLS:PRINT"            ENTERING A PATTERN"
2950 PRINT"                ---------------"
2960 PRINT
2970 PRINT" You must enter a pattern of live frogs"
2980 PRINT"on the next screen."
2990 PRINT:PRINT
3000 PRINT"  To produce a frog press the spacebar"
3010 PRINT"then move the cursor using the arrow"
3020 PRINT"keys and press the spacebar again "
3030 PRINT"for your next frog. To delete a "
3040 PRINT"frog use the 'DELETE' key."
3050 PRINT:PRINT
3060 PRINT"  Start with a simple pattern, such as"
3070 PRINT"a line, square or diamond."
3080 PRINT:PRINT
3090 PRINT"  When you're happy with your pattern"
3100 PRINT"then press the 'RETURN' key."
3110 PRINT:PRINT:PRINT:PRINT:PRINT
3120 PRINT" Press spacebar to continue"
3130 key=GET
3140 CLS
3150 REM Enable auto key repeat.
3160 *FX12,0
3170 ENDPROC
3180
3190 DEF PROCPROMPTS
3200 CLS
3210 PRINT TAB(0,0);"Enter pattern (spacebar & arrow ke
ys).";
3220 PRINT TAB(0,31);"Press H for help, RETURN key to R
UN.";
3230 PRINT TAB(20,15);
3240 ENDPROC
3250
3260 DEF PROCLOGO
3270 LOCAL x%,y%
3280 FOR i%=1TO8:READ note(i%)
3290 NEXT i%
3300 FOR i%=1TO30:READ x%,y%
3310 PROCBLOCK(x%,y%)
3320 SOUND 1,-15,note(RND(8)),4
3330 PROCWAIT(20)
3340 NEXT i%
3350 PROCWAIT(20)
3360 SOUND 1,-15,101,40
3370 SOUND 2,-15,117,40
3380 SOUND 3,-15,129,40
3390 PROCWAIT(200)
3400 DATA 5,13,21,25,33,41,49,53
3410 DATA 21,11,19,9,7,9,7,11,15,7
3420 DATA 15,11,29,11,27,15,19,15,15,15
3430 DATA 11,15,19,7,19,13,31,7,29,7
3440 DATA 27,7,27,11,29,15,31,15,15,9
3450 DATA 7,13,9,15,23,7,15,13,27,9
3460 DATA 7,7,21,7,7,15,27,13,19,11
3470 ENDPROC
3480
3490 DEF PROCBLOCK(x%,y%)
3500 VDU 31,x%,y%
3510 VDU 224;224;8,8,10,224;224;
3520 ENDPROC
3530
3540 DEF PROCWAIT(t%)
3550 TIME=0
3560 REPEAT
3570 UNTIL TIME=t%
3580 ENDPROC
```

INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO INFO

SONY "PROFEEL" COLOUR TV/MONITOR

David Willington of Harrow writes to tell us that he has connected his BBC micro to use his Sony Profeel monitor. The connector required is the same as for the disc interface (34 pin) and is obtainable complete with ribbon cable from Watford Electronics amongst others. The problem is in the numbering scheme used on these connectors which is different on the Sony from most other IDCs.

BBC RGB DIN	Function	34 WAY CONN TO VDU Sony no	Equivalent no on non-Sony IDC
5	0v	8	20
-	+12v	19	31
3	blue	27	15
2	green	26	17
1	red	25	19
4	sync	30	9
-	mode switch	33	3

Note - connect 10kohms between Sony pin numbers 19 and 33.

Program tested on O.S. 0·1 and 1·2

PROCEDURE FUNCTION LIBRARY

This month we give a function that validates a date in the form DD/MM/YY although other characters can replace the / for example DD.MM.YY or DD-MM-YY would also be accepted.

The lines 10-80 demonstrate the use of the function, and the function itself starts at line 1000. The variables day, mth, yr have purposely not been made LOCAL to the procedure so that you can use their values in the main program.

Beware, because the function contains a DATA statement, and if your program has DATA statements too it would be adviseable to RESTORE to your data just in case (particularly if your DATA statements appear later in the program than line 1120.

```
   10 the_cows_come_home=FALSE
   20 REPEAT
   30   REPEAT
   40    INPUT"Date in the form
DD.MM.YY ",todays_date$
   50   UNTIL
FNvalid_date(todays_date$)
   60   PRINT"Date validated
successfully"
   70 UNTIL the_cows_come_home
   80 END
   90
 1000 DEF FNvalid_date(date$)
 1010 LOCAL month,ndays_in_mth,sep
 1020 RESTORE 1120
 1030 sep=ASC(MID$(date$,3,1)):IF
sep<32 OR sep>47 THEN 1140
 1040 sep=ASC(MID$(date$,6,1)):IF
sep<32 OR sep>47 THEN 1140
 1050 day=VAL(LEFT$(date$,2))
 1060 mth=VAL(MID$(date$,4,2))
 1070 yr= VAL(MID$(date$,7,2))
 1080 IF day<1 OR mth<1 OR yr<0 THEN
1140
 1090 FOR month=1 TO mth:READ
ndays_in_mth:NEXT
 1100 IF yr MOD 4=0 AND mth=2
ndays_in_mth=29
 1110 IF day>ndays_in_mth THEN 1140
 1120 DATA
31,28,31,30,31,30,31,31,30,31,30,31
 1130 =TRUE
 1140 =FALSE
```

by Sheridan Williams

DISC HINTS DISC HINTS DISC HINTS DISC HINTS DISC HI

DISC SPACE ECONOMIES

On disc systems, PAGE is automatically set to &1900. On cassette systems it is &E00. This means that you have more than 3K less memory in which to run programs on a disc based machine. Last month we gave a 'move-down' program to help get around this problem. But there is a better way around the problem in certain circumstances. If your program does not load or save DATA to disc (ie. if it does not use the PUT and GET byte commands), then the disc operating system only uses part of the buffer reserved for it. This means that you can set PAGE to &1200 (ie. PAGE=&1200 <return>) before using the computer (though remember not to press Break), and you will save 1792 bytes of memory.

[We have been testing this for some time, and have so far encountered no problems. Ed.]

COLOURED TITLES

John Yale has sent in a hint which will enable you to highlight disc titles by using Teletext colour codes.

Whilst coloured filenames are not possible on the disc, the disc title may be made coloured to make it stand out when a *CAT is done. Just insert the teletext colour code by using SHIFT and one of the function keys. You can also insert graphics characters in the same way. See "What to do on 1.2" in this issue for further details.

MICRO SKETCH (A SIMPLE LINE DRAWING PROGRAM) (16k)

Program tested on O.S. 0.1 and 1.2

by J C Fenton

How small can a 'drawing' program be while still remaining useful and fun? Answer - just one line! The single line program given below enables straight lines to be drawn between any two specified points on the screen, and some extremely good sketches can be obtained.

When the program is RUN the screen will be cleared and the program will wait until a key is pressed (eg. <space>). After being 'started' in this way a display is given, in the top left hand corner of the screen, of the co-ordinates of a small 'dot' cursor. The cursor is moved and lines are drawn by using the following keys:

MOVE COMMANDS		OTHER COMMANDS	
'<'	move left	'F'	record 'from' position
'>'	move right	'T'	draw 'to' this position
'A'	move up	'E'	'end' the session
'Z'	move down		

Drawn lines are deleted by merely passing the cursor over them. The effect of this feature is to force a 'hole' in a line if you have to come out of a closed shape. It is a simple enough matter to fill the hole back in by drawing a line which is only one 'position' size in length.

```
10MODE4:X%=400:Y%=400:REPEAT:G=GET:PLOT71,X%,Y%:X%=X%+4*(G=44)-4*(G=46):Y%=Y%+4*(G=9
0)-4*(G=65):MOVEA%,B%:PLOT69+64*(G=84),X%,Y%:A%=A%+(A%-X%)*(G=70):B%=B%+(B%-Y%)*(G=7
0):VDU4,30:PRINT;X%;",";Y%;"  ";:VDU5:UNTILG=69:MODE7
```

POINTS ARISING

SIMPLE MUSICAL KEYBOARD (BEEBUG no.9)

The 'simple musical keyboard' program submitted by J C Fenton may be improved by resetting the auto-repeat delay with *FX 11,5 and changing the duration of the note (the fourth parameter of the sound function) from 10 to 2. The slight 'blip' in the note after 1 second is eliminated and the same note can now be played repeatedly very quickly.

EPSON SCREEN DUMP

1) PUSH BUTTON DUMP

There is an easy way to activate the Epson Screen Dump (BEEBUG no.9 p.10) during the running of a Basic program to print out the screen at any given point. The method was described in connection with the Seikosha, but not the Epson. It involves using the ON ERROR call. Simply make the first line of the program producing the screen: 1 ON ERROR GOTO 10000 At line 10000 put the call to the screen dump, and press <Escape> when you want a copy of the screen. You must however make sure that the screen producing program is error-free, otherwise you will get a dump each time an error is encountered. To get around this you could make the printout call dependent on ERR=17, ie. 10000 IF ERR=17 THEN CALL...

2) DISC CALL

When using the Epson Dump from disc, if you have saved it with *SAVE "SDUMP" A00 B00 A00 then you can automatically load it from disc AND run it using the call *SDUMP, (provided that SDUMP is in the disc library area - see '*LIB' in Disc Filing System article in this issue). SDUMP behaves in this way like a new command, since it is called in a similar way to other DFS commands, eg. *COMPACT etc. Do not call it *DUMP since this is a reserved word.

To produce a screen dump from a program when you press <Escape>, just add the single line: 1 ON ERROR VDU2:*SDUMP

SOFTWARE UPDATE SOFTWARE UPDATE SOFTWARE UPDATE

In this column we bring information and ideas about programs in our software library, including the Wordwise word processor.

Wordwise continues to be very popular among BEEBUG members, especially now that we have been able to offer it with a 1.2 Operating System. And for the past two issues of BEEBUG, the whole magazine has been produced using Wordwise. This month we give a useful routine for those using Wordwise in conjunction with an Epson printer.

WORDWISE

CONFIGURING WORDWISE FOR EPSON PRINTERS

Wordwise is probably the most popular word processor, and the Epson is probably the most popular printer for use with the BBC micro. I have used such a configuration for some time. The program below runs from Basic, and sets the user keys to produce a series of extremely useful text format commands. It then calls Wordwise. If you have a disc system you may wish to make this program auto-boot (see BEEBUG no.9 p.19).

The keys set are as follows:

f0	f1	f2	f3	f4	f5	f6	f7	f8	f9
hash	u/line on	u/line off	emph on	emph off	cond on	cond off	enlarge on	enlarge off	standard page

For example, if you are entering text in Wordwise and you wish to make the following text enlarged then press 'shift-control-f7' you will see a green 0C14 appear, this is the command that the Epson needs to produce enlarged text.

The 'hash sign' option needs further explanation, as it requires that you have already set the switches inside the Epson for the British character set; see the Epson manual for details. On pressing shift-control-f0 a long green command (0C27,82,0,35,27,82,3) appears which switches back the USA character set, prints a hash, and then switches back to British again.

Finally the f9 key is set up for what I find the most useful starting configuration. See the back of the Wordwise manual for explanations.

```
10 REM Wordwise configure for Epson
20
30 REM Auto line feed at end of line
40 *FX6,0
50 REM Hash sign
60 *KEY0 |!!0C27,82,0,35,27,82,3|!"
70 REM Underline on
80 *KEY1 |!!0C27,45,1|!"
90 REM Underline off
100 *KEY2 |!!0C27,45,0|!"
110 REM Emphasised on
120 *KEY3 |!!0C27,69|!"
130 REM Emphasised off
140 *KEY4 |!!0C27,70|!"
150 REM Condensed on
160 *KEY5 |!!0C15|!"
170 REM Condensed off
180 *KEY6 |!!0C18|!"
190 REM Enlarged on
200 *KEY7 |!!0C14|!"
210 REM Enlarged off
220 *KEY8 |!!0C20|!"
230
240 REM Default page settings
250
260 *KEY9 |!!LM5|M|!!LL76|M|!!PL72|M
|!!TS2|M|!!BS2|M|!!EP|M|!!DP35|M|!!JO|M
|!!DT4,10,15,20,25,30,35,40,45|M|M
270 *WORDWISE
```

SW

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

WORDWISE PACKAGE PLUS NEW 1.2 ROM IS OFFERED AT £45.00 INCLUDING P&P & VAT. 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.

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 relate 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 Rob Pickering, John Yale, Adrian Calcraft, Tim Powys-Lybbe, and Graham Greatrix for assistance with this issue.

BEEBUG MEMBERS SOFTWARE LIBRARY. NEW TITLES.

APPLICATIONS 3 – ARTIST 2 £4.50 inc VAT + 50p. pp.
(32k with joysticks)

This program is a developement of that which appeared in the Dec/Jan issue. Many members wrote in to say how much they enjoyed this program, and this new version adds many of the features which were requested.

The program allows pictures to be drawn on a Mode 2 screen in eight colours using the BBC joysticks to position Lines, Triangles, Rectangles, Circles or just to doodle. All function selection is by positioning the cursor in the relevent menu box displayed at the top of the screen and pushing the fire button, so there are no key functions to remember, all options being on view continuously.

Functions provided are: Colour, Shape, Filled/Unfilled, Clear screen, Save to disc or tape, Plotting logic, and Scaled mode. Plotting logic corresponds to the first GCOL parameter and in Exclusive OR mode allows complex patterns to be drawn very quickly. Scaled mode restricts the cursor movement to any selected part of the screen allowing fine details to be drawn.

A status display is also maintained on the screen, showing the current colour, shape, plot mode, and cursor co-ordinates. This last feature allows the cursor to be very accurately positioned on the screen especially when used in conjunction with the scaled mode. A full instruction manual is provided, including configuration details for using non-standard joysticks.

This drawing package will be found to compare favourably with programs costing many times the BEEBUG price of £4.5Ø. The cassette contains both a cassette and a disc version.

UTILITIES 2 – EXMON (16k) £6.90 inc VAT + 50p. pp.

EXMON is an extended machine code monitor for the BBC Micro. It was developed out of the winning program of BEEBUG's second software competition. A vast number of extra facilities have been added to the original entry, making EXMON far more than an ordinary monitor. The BBC micro is an extraordinarily advanced machine. EXMON has been designed to complement it, and provides 3Ø new commands, all achieved in machine code. It is a must for anyone using machine code or assembler on the Beeb.

The program occupies about 5k bytes and may be located anywhere in memory using one of its built in commands. When activated it displays the 65Ø2 register contents at the top of the screen, and any selected portion of memory may be displayed in Hex, ASCII or disassembled mnemonics. The register or memory contents may be changed, and program execution started or single stepped from the current program counter position until a breakpoint is reached (up to 5 breakpoints may be set at any time). Whilst single stepping, the next instruction is disassembled into mnemonic form and the register display updated.

Some of the more unusual features of this Monitor are the ability to relocate programs in memory (including itself !),ie EXMON can move a section of machine code, and adjust the code to run at the new location. EXMON accesses the symbol table of the current BASIC program, so that when debugging an assembler program it is not necessary to know the actual addresses within that program but only the names of the labels and data areas used. Arbitrary BASIC type expressions may also be used in many of the commands to calculate addresses etc.(except on O.S Ø.1). The cassette contains both a cassette and disc version.

FOR ORDERING INFORMATION SEE p.2Ø OF THIS MONTHS SUPPLEMENT.

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