Vol.12 No.7 December 1993

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

- BEEBART
- PYRAMID PATIENCE
- RELOCATOR
- PUTTING DIRECTORIES TO WORK

BEEBUG Vol.12 No.7 December 1993

FEATURES

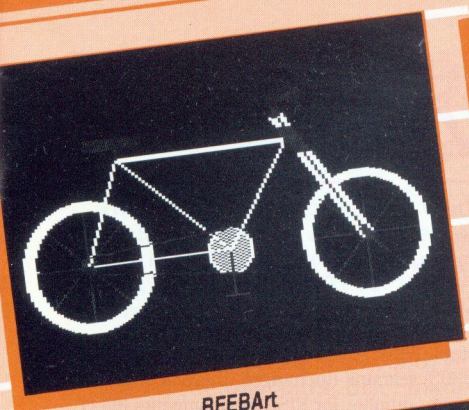
REGULAR ITEMS

HINTS & TIPS

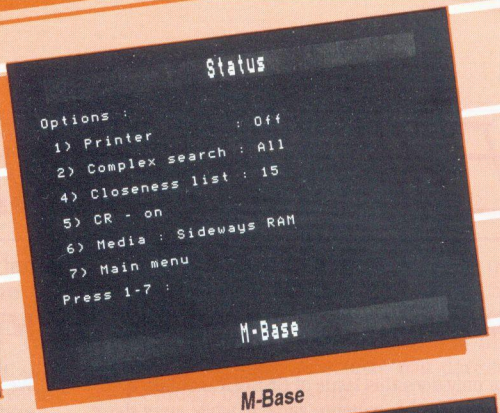
PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

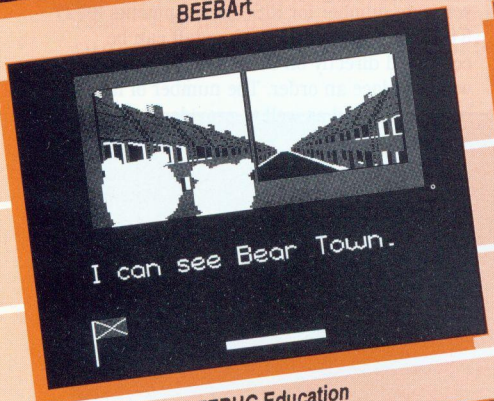
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



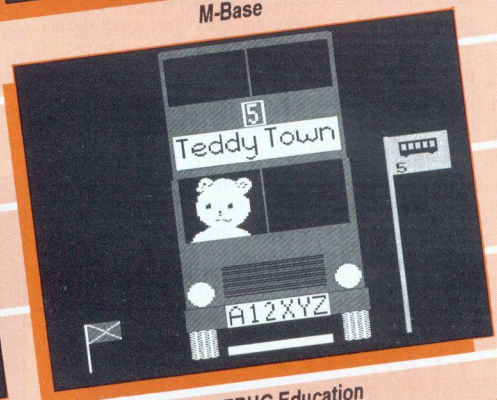
BEEBArt



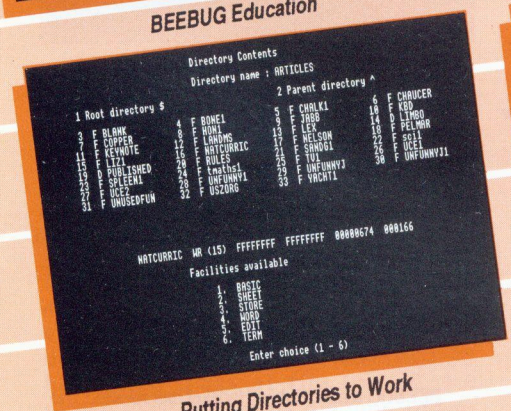
M-Base



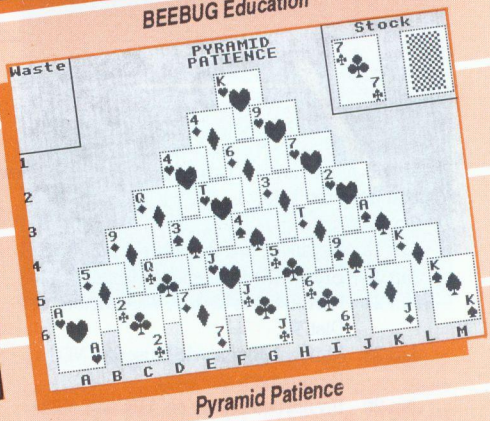
BEEBUG Education



BEEBUG Education



Putting Directories to Work



Pyramid Patience

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

FUN AND GAMES FOR CHRISTMAS

Not only does this issue of BEEBUG contain a highly playable patience game to while away those quiet moments during the holiday, but we have included an updated version of the very popular Robol game on the magazine disc, and an updated calendar program ready for the new year. The magazine (and disc) also contains a major new item in the form of a very comprehensive painting program. With all that and much more in this month's magazine it is clear that there is still much interest in the BBC micro.

NEW PHONE LINES

We have recently installed a new telephone system. While the original number (0727 840303) remains unchanged, there is now a separate number for RISC Developments (0727 843600), and Beebug's sales staff can be contacted directly on 0727 840305 when you want to place an order. The number of lines is being increased as well to provide a better service.

Mike Williams

News

ALL FORMATS COMPUTER FAIRS

Details of All Formats Computer Fairs for the new year are given below:

Jan 15 Haydock Park Racecourse (J23 M6).
Jan 16 Brunel Centre, Temple Meads, Bristol.
Jan 22 Northumbria Centre, Washington Dist.12.
Jan 23 National Motorcycle Museum, NEC, Birmingham (J6 M42).
Jan 29 Brentwood Centre off A12, J28 M25.
Feb 12 Haydock Park Racecourse (J23 M6).
Feb 13 National Motorcycle Museum, NEC, Birmingham (J6 M42).

Admission costs £4 for adults (£2 after 2pm), £2 for children, wheelchair users free. For 50 £1 off vouchers send a stamped addressed envelope to Bruce Everiss, Maple Leaf, Stretton-on-Fosse, Moreton-in-Marsh, Gloucestershire GL56 9QX, or telephone 0608 662212.

BETT '94

The 1994 *British Education for Training and Technology* exhibition will be held at the National Hall, Olympia, London from 12th to 15th January 1994. This show is a must for anyone with an interest in computers and education, and there is sure to be a strong presence from the Acorn world. RISC Developments Ltd, publishers of RISC User, will be there on stand 402.

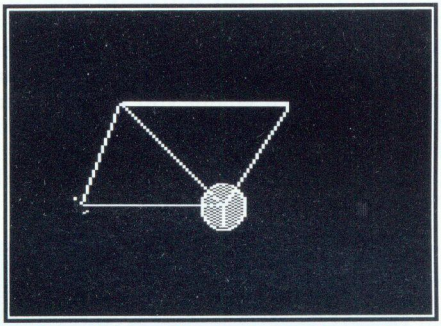
THE BEEB SCRAPBOOK

BEEBUG reader Ruben Hadekel has released *The Beeb Scrapbook*, a collection of articles and programs for users of the BBC micro. The articles include material on programming techniques, and instructions for the programs, including items previously published in BEEBUG under agreement with RISC Developments. The disc, in ADFS format for the Master 128, costs £14.50 inclusive direct from Ruben Hadekel, 4 Lalor Street, London SW6 5SR, tel. 071 736 5429.

BEEBArt

Budding Van Goghs, lend an ear to Mark Brading.

This program was originally written to replace *Timpaint*, as supplied on the original Master welcome disc. For maximum flexibility I wrote two versions, one to run in screen mode one, and one to run in mode two. The mode two version is listed below, both versions are on the magazine disc, loaded via *Bload*. Note that these programs will only run on a Master 128 and Master Compact.

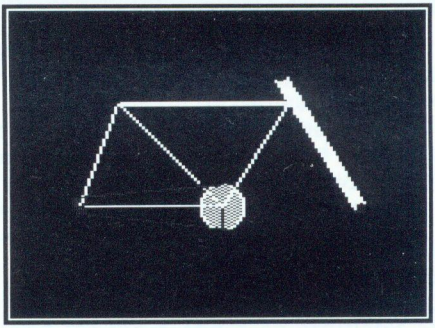


The pictures illustrate the stages in drawing a bicycle

USING THE PROGRAMS

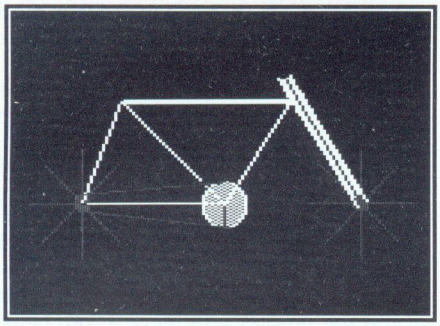
I originally designed *Beebart* to be easier to use than Timpaint; I'm not sure it turned out that way. However, it is definitely quicker, at least when it comes to selecting tools and colours. Rather than have a visual toolbox which reduced the useful drawing area, I decided to have two lines at the top of the screen used for status display, and to have all the tools selectable by logical keystrokes.

When the program is first run the pointer, a white triangle, is in the centre of the screen, and its position is shown at the top. The pointer can be moved around with the Cursor keys. By default it moves in steps of 9 OS units, but this can be changed using Shift with the Up and Down Cursor keys. You start off in freehand draw mode, with no colour selected. You can select a colour by pressing a number key in the range 1 to 8 (1 to 4 in the mode 1 version). These represent the logical colour numbers apart from 8 (or 4) which selects black. Pressing key 0 selects no colour. When you select a colour by pressing a number, the text at the top of the screen will turn the colour you have selected (for no colour and black the text will remain white) and the pointer will leave a trail behind it when you move it around. If no colour is selected then the pointer leaves no trail.



If you wish to use one of the Master's patterns instead of a block colour, then this can be done by pressing Shift in combination with one of the function keys. When you begin drawing, you will find that the selected colour no longer has any effect, and the selected pattern is used instead. Pressing Shift in combination with '@' returns you to normal use of colour. Patterns may not be easy to spot when used with single lines, but they can be easily seen in filled shapes and flood fills.

For the technical amongst you, the pattern selection works by multiplying the number you press in combination with Shift by 16, and using the result as the value before the comma in all future GCOL statements.



In the mode one version of the program, there is a additional feature which allows you to change the standard black, red, yellow, white palette with any of the 8 colours in the mode 2 palette. To do this, select the colour that you want to change (press 1, 2, 3, or 4) and then press 'C'. You will then be prompted for the colour number of the colour you want to change it to. Press the colour number of the colour you want and it will immediately replace the previously selected colour, taking over that colour's number.

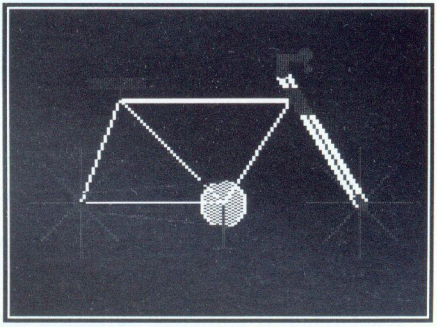
THE TOOLS

To use the tools press 'T' while in the freehand drawing mode. You will be prompted to enter one of the following letters: A, C, E, F, L, P, Q, R, S, T. Their effect is as follows:

A - Arcs.
C - Circles, filled or outline.
E - Ellipses, filled or outline.
F - Flood fill.
L - Lines at any angle.
P - Parallelograms, filled only.
Q - Quit, return to freehand drawing.
R - Rectangles, filled only.
S - Segments of circles.
T - Text anywhere on the screen.

Only the text tool will return you to freehand drawing immediately. All the others will prompt you to press 'Q' to quit after they have been used once. Pressing Q at this stage returns you to freehand drawing. In most cases pressing any other keys, including the Cursor keys, allows you to use the same tool again.

The tools fall into two main groups. Those that require two points to make them work, and those that require three. Those that need two are circles, rectangles, and lines. When you select any of these you will be prompted to press Return to set point one - circles will first prompt you to select filed or outline - move the pointer around as usual and press Return where you want the first point to be. With circles this is the middle of the circle, with lines it is one end of the line, and with rectangles it one of the corners of the rectangle.



Now move the cursor around and you will see the shape change size as you do so. When the shape is the size and in the position you want, press Return and it will be drawn in the current colour or pattern. Colours or patterns can be selected at any time during the use of any tool. If you decide, while using a tool, that you do not want to use it after all, or that the first point is in the wrong place, simply press 0 to select no colour and finish the tool's

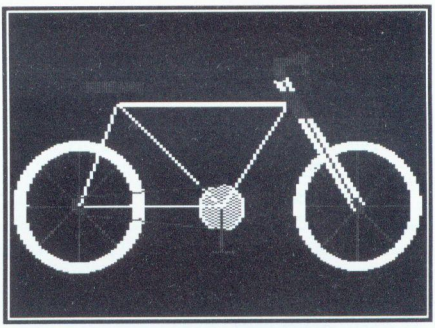
sequence without moving the pointer. This will prevent the tool from having any effect and your picture will remain unchanged. This is not available in flood fill or text.

The use of tools that require three points is very similar, just requiring one more step. The tools that require three points to be set are ellipses, parallelograms, arcs, and segments. When any of these tools are selected you will, as before, be prompted to press Return to set the first point (except in ellipses, where you will first need to select filled or outline). Pressing Return will lock the first point of your shape. In ellipses this is the middle of the ellipse, in parallelograms it is the first corner of the parallelogram, in arcs and sectors it is the centre of the circle that the arc or segment is going to form part of. When the first point is fixed, you will be prompted to press Return to lock the second point.

Moving the pointer around you will see the shape grow. In ellipses you will only see an effect if you move horizontally away from your centre point. During this a line will be drawn in both directions away from your centre point. This line will form one of the axes for your ellipse. At this stage you cannot rotate the axes. In parallelograms you are now fixing the second axis of your parallelogram, this can be at any angle you like from the first one.

In arcs and segments you are fixing a point on the radius of the circle. Unlike whole circles, where the position of the point on the radius did not matter so long as it was on the radius, the point must be positioned not only on the radius of the circle, but also where you want the arc/segment to start. The best way for you to find out how to do it is to try it for yourself. When the second point is fixed you will see that you are now moving the final point of the shape; it only remains to fix the third point with Return and lock the shape.

With ellipses the third point is the vertical axes and can be put at any angle. With parallelograms it is the third corner of the parallelogram, and in arcs and sectors it is the other end of the arc/sector. Once again, experimentation is really the only way to get this clear in your mind.



The flood fill tool is very simple; select it by pressing 'F' in the tool menu and then press Return in an empty space to flood it with the currently selected colour of pattern. You can press 'Q' to quit at any time.

Text is equally simple; when the text option is selected from the tool menu you will be prompted for a text string, and for the X and Y co-ordinates at which the text will be plotted. Note that these are not graphics co-ordinates, but are equivalent to the TAB(X,Y) statement. Text can be printed in any colour (though patterns cannot be used) and the colour must be selected before the text facility is entered - colours can not be selected from within the procedure.

SAVING AND LOADING

To save or load a picture type S or L respectively when in freehand draw mode. You will be prompted for a

filename, and the picture will be saved to or loaded from that filename on the current drive. Pathnames can be entered, and quotation marks are not required. Now, get drawing.

```
   10 REM Program Beebart
   20 REM Version B 1.0
   30 REM Author  Mark Brading
   40 REM BEEBUG  December 1993
   50 REM Program subject to copyright
   60 :
  100 MODE 2
  110 PROCinit
  120 PROCdraw
  130 END
  140 :
 1000 DEFPROCdraw
 1010 REPEAT
 1020 PROCsetpointvar
 1030 PROCprintpos("Press 'T' for tools.
")
 1040 PROCpointer
 1050 PROCcurkey
 1060 *FX21,0
 1070 GCOLpattern%,col%
 1080 IF tools% THEN PROCtools
 1090 IF load% THEN PROCloader
 1100 IF save% THEN PROCsaver
 1110 IF draw% THEN DRAW x%,y% ELSE MOVE
 x%,y%
 1120 UNTIL end%
 1130 ENDPROC
 1140 :
 1150 DEFPROCcurkey
 1160 REM
 1170 keypress% = GET
 1180 PROCpointer
 1190 IF INKEY(-1) THEN GOTO 1260
 1200 IF keypress% = 48 THEN draw% = FAL
SE
 1210 IF keypress% >48 AND keypress% <57
 THEN col% = keypress%-48:draw% = TRUE
 1220 IF keypress% = 136 THEN x%=x%-step
%:IF x%<10 THEN x% = 10
 1230 IF keypress% = 137 THEN x%=x%+step
%:IF x%>1269 THEN x% = 1269
 1240 IF keypress% = 138 THEN y%=y%-step
%:IF y%<4 THEN y%=4
 1250 IF keypress% = 139 THEN y%=y%+step
%:IF y%>943 THEN y% = 943
 1260 IF keypress% = 139 AND INKEY(-1) T
HEN step% = step% + 4:IF step%>200 THEN
step% = 200
 1270 IF keypress% = 138 AND INKEY(-1) T
HEN step% = step% - 4:IF step%<1 THEN st
ep%=1
 1280 IF keypress% >32 AND keypress% < 3
7 AND INKEY(-1) THEN pattern% = (keypres
s%-32)*16
 1290 IF keypress% = 64 AND INKEY(-1) TH
EN pattern% = 0
 1300 IF keypress% = 84 OR keypress% = 1
16  THEN tools% = TRUE
 1310 IF keypress% = 13 THEN return% = T
RUE
 1320 IF keypress% = 9 THEN undo% = TRUE
 1330 IF keypress% = 81 OR keypress% = 1
13 THEN end% = TRUE
 1340 IF keypress% = 83 OR keypress% = 1
15 THEN save% = TRUE
 1350 IF keypress% = 76 OR keypress% = 1
08 THEN load% = TRUE
 1360 ENDPROC
 1370 :
 1380 DEFPROCinit
 1390 *FX4,1
 1400 col%=0:x%=640:y%=512:draw%=FALSE:e
nd%=FALSE:tools%=FALSE:step%=9:VDU19,8,0
,0,0,0
 1410 pattern%=FALSE:return%=FALSE:undo%
=FALSE:load%=FALSE:save%=FALSE:PROCborde
r
 1420 *FX151,11,0
 1430 *FX151,0,1
 1440 ENDPROC
 1450 :
 1460 DEFPROCtools
 1470 tools% = FALSE
 1480 PROCprintpos("A,C,E,F,L,P,Q,R,S,T.
")
 1490 PRINTTAB(0,0)"Select tool.
":REM 8 spaces
 1500 keypress% = GET
 1510 IF keypress% = 67 OR keypress% = 9
9 THEN PROCcircle
 1520 IF keypress% = 69 OR keypress% = 1
01 THEN PROCellipse
 1530 IF keypress% = 76 OR keypress% = 1
```

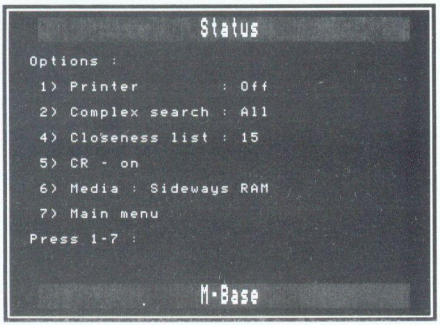
```
08 THEN PROCline
 1540 IF keypress% = 82 OR keypress% = 1
14 THEN PROCrectangle
 1550 IF keypress% = 70 OR keypress% = 1
02 THEN PROCflood
 1560 IF keypress% = 65 OR keypress% = 9
7 THEN PROCarc
 1570 IF keypress% = 80 OR keypress% = 1
12 THEN PROCpar
 1580 IF keypress% = 83 OR keypress% = 1
05 THEN PROCsegment
 1590 IF keypress% = 84 OR keypress% = 1
16 THEN PROCtext
 1600 ENDPROC
 1610 :
 1620 DEFPROCflood
 1630 PRINTTAB(0,0) SPC 40
 1640 PRINTTAB(0,0)"Flood Fill.
":REM 9 Spaces
 1650 PROCwait(5000)
 1660 REPEAT
 1670 PROCsetpointvar
 1680 PROCprintpos("Press 'Q' to quit.
"):REM 2 spaces after the full stop
 1690 PROCpointer:PROCcurkey:*FX21,0
 1700 GCOLpattern%,col%
 1710 IF return% THEN PLOT133,x%,y%:retu
rn%=FALSE
 1720 IF undo% THEN PROCundo
 1730 UNTIL end%
 1740 end%=FALSE
 1750 ENDPROC
 1760 :
 1770 DEF PROCundo
 1780 VDU19,128,col%,0,0,0
 1790 GCOL0,8:PLOT133,x%,y%:VDU19,128,12
8,0,0,0
 1800 undo%=FALSE:GCOLpattern%,col%
 1810 ENDPROC
 1820 :
 1830 DEF PROCcircle
 1840 PRINTTAB(0,0) SPC 40
 1850 REPEAT
 1860 PROCprintpos("Press 'Q' to quit.
"):REM Two spaces
 1870 PRINTTAB(0,0) "(F)ill or (O)utline
?"
 1880 PROCsetpointvar:PROCpointer:PROCcu
rkey
 1890 IF keypress% = 70 OR keypress% = 1
02 THEN PROCshape("Filled Circles.     "
,157,5000):REM 5 Spaces
 1900 IF keypress% = 79 OR keypress% = 1
11 THEN PROCshape("Outline Circles.    "
,149,5000):REM 4 Spaces
 1910 UNTIL end%
 1920 end%=FALSE
 1930 ENDPROC
 1940 :
 1950 DEF PROCshape(name$,code%,dely%)
 1960 PRINTTAB(0,0)name$:PROCwait(dely%)
 1970 REPEAT
 1980 PROCsetpointvar:PROCprintpos("RETU
RN to set P one.")
 1990 PROCpointer:PROCcurkey:*FX21,0
 2000 IF return% THEN x1%=x%:y1%=y%
 2010 UNTIL return%
 2020 return%=FALSE
 2030 REPEAT
 2040 x2%=x%:y2%=y%:PROCsetpointvar
 2050 PROCprintpos("RETURN to lock shape
")
 2060 MOVE x1%,y1%:PLOT(code%+1),x2%,y2%
 2070 PROCpointer:PROCcurkey:*FX21,0
 2080 MOVE x1%,y1%:PLOT(code%+1),x2%,y2%
 2090 UNTIL return%
 2100 return%=FALSE:keypress%=0
 2110 GCOLpattern%,col%:MOVE x1%,y1%
 2120 IF draw% PLOTcode%,x2%,y2%
 2130 ENDPROC
 2140 :
 2150 DEF PROCline
 2160 PRINTTAB(0,0) SPC 40
 2170 PROCshape("Line.                 ",5
,5000):REM 15 Spaces
 2180 REPEAT
 2190 PROCsetpointvar:PROCpointer
 2200 PROCprintpos("Press 'Q' to quit.
"):REM 2 Spaces
 2210 PROCcurkey:*FX21,0
 2220 IF NOT end% THEN PROCshape("Line.
            ",5,0):REM 15 Spaces
 2230 UNTIL end%
 2240 end%=FALSE
 2250 ENDPROC
 2260 :
 2270 DEF PROCrectangle
 2280 PRINTTAB(0,0) SPC 40
 2290 PROCshape("Filled Rectangles.  ",1
```

continued on page 34

M-Base (Part 2)

Ian Palmer continues the development of his comprehensive database.

This is the second part of the M-Base program. First you need to type the lines of the program which accompany this text. These lines should be added to part 1 given in the previous issue. This will give you a fully working program, apart from the search routines which will be given in part 3. You can now use the program as follows.



The Status Menu

DELETING RECORDS

Before deleting a record you need to find out its number as stored in the database. This is shown each time that record is displayed, either in the viewing of records or during a search.

Once you have found the number, select option 3 of the main menu. You will then be asked to enter that number and the record corresponding to that number will be displayed for you to confirm that it is the correct one. If it is just press 'Y', otherwise press 'N' or Escape.

Every time you delete a record the numbers of other records may change, so you need to check the numbers of any other records you wish to delete. This is due to the automatic housekeeping of the program. In general only one record will change place for each record deleted.

SAVING A DATABASE

To save the database that is in memory, select option 7 of the main menu. You will then be asked to enter the filename you wish to use. In fact, if you wish to use the name entered when you created the database just press Return, otherwise you need to press Ctrl-U to clear that name and enter the name you wish to use.

LOADING A DATABASE

To load a database into memory select option 6 of the main menu, and enter the filename of the database at the prompt.

VIEWING INFORMATION ABOUT A DATABASE

Select option 5 of the main menu and you will be shown the current database's name, the number of fields it has, the number of records stored and the name of each field. Press Space to return to the main menu.

THE STATUS MENU

This menu allows you to change certain characteristics of M-Base. To get to this menu select option 9 of the main menu. The status menu has seven options as described below.

1. Printer on/off - this option toggles the state of the printer flag. If the flag is *On* any records displayed on the screen will also be sent to the printer. The default is *Off*.

2. Complex search all/any - this option toggles the state of this flag between all and any. The meaning of this flag is described later with the complex search.

3. Print options - this allows you to specify which fields are sent to the printer when the printer flag is set *On*. When you select this option you will be taken to a screen similar to an empty record, only in this one you should enter the words *On* or *Off* in each field. In fact, the old states will be placed in the fields when you reach them so, to keep the old states, just move onto the next.

4. Closeness list - this option allows you to enter the length of the list produced by the closeness search. To change the value select this option and enter the new value. Default 15.

5. CR On/Off - this is another *On Off* flag, again related to the printer. When this flag is on each field printed is printed on a line of it's own. If the flag is off, more than one field may be printed on each line.

6. Storage media Disc/Sideways RAM - this option allows you to specify which storage media you wish to use. This option needs confirmation, by pressing 'Y'. If you were using Sideways RAM and have a file in memory you will be asked if you wish to save it before changing to disc. If you were using disc and have a file open it will automatically be closed.

7. Return to main menu.

USING DISC STORAGE

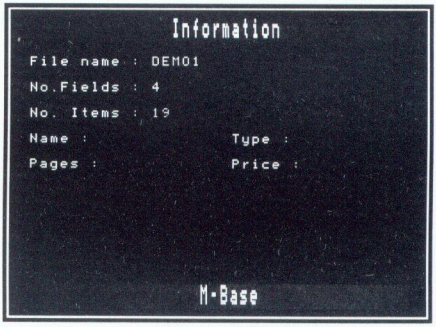
When using disc rather than sideways RAM the only difference in usage, other than speed, is that options 6 & 7 are renamed, and perform slightly different functions. Option 6 becomes 'Open file' and opens a file for use. Option 7 becomes 'Close file' and closes the current file; it does not ask you for a file name.

STATUS SETUP FILE AND MANUAL SETTING

The above status flags can be set automatically, except the storage media, via a setup file called MSetUp. The file should made up of commands of the following format.

```
[name] command : state
```

where *name* is the name of the database which the command is to refer to, or '*' for all databases.



The Information Screen

command should be one of the following, with state being one of those given for each command.

CR - sets the state of the CR flag. For off state is 0, off or OFF, for on state is 1, on or ON.

PRINTER - sets the state of the printer flag. States are the same as for CR.

COMP - sets the state of the complex search flag, states are any of : ANY any ALL all.

LENGTH - sets the length of the closeness list, state is any number from 3 to 50.

PRINT - sets the print states of the fields, state is a list of 1's and 0's, one for each field, 1 for On, 0 for Off.

All commands should be in capital letters, any spaces in the command line are ignored. The MSetUp file is executed by typing '*MX' on the main menu.

The above commands can also be typed direct from the keyboard by pressing '+' on the main menu. In this case you should not type the [name] part as this is added automatically.

An MSetUp file might look like this: Assume you have four databases given below, the number in the brackets after each name is the number of fields that database has.

Address(7), Tapes(3), Videos(4), CDs(3)

MSetUp file :

```
[Address] PRINT : 1111110
[tapes] PRINT : 100
[CDS] PRINT : 101
[videos] PRINT : 1 1 0 0
[*] CR : off
[*] PRINTER : off
[*] LENGTH : 25
[videos] LENGTH : 20
[*] COMP : any
```

As can be seen, the case of the name part is not important. As commands are executed in the order given, the two LENGTH commands are valid in that order, if they were in the other order the [*] command would overwrite the [videos] command.

Any errors will be reported on execution. The set up is immediate and thus if you change database you will need to re-run the MSetUp.

EXIT FROM THE PROGRAM

You should exit from the program using option 0 of the main menu rather than pressing Break or typing *BASIC. This is to ensure that the database is closed if using M-Base in the disc option, and that the 'Grab old database' option works properly. This option can be found on the 'New / alter database' menu (option 4). This allows you to exit M-Base and re-enter it after running another program without needing to re-load the database you were using, as long as any other programs used do not interfere with sideways RAM.

That concludes part 2 of M-Base. Part 3 will supply you with three search routines, including the unique 'Closeness' search.

```
  10 REM Program M-BASE-2
  40 REM BEEBUG  December 1993
3170 REM ***** Delete Records *****
3180 :
3190 DEFPROCdelete:LOCAL f%,b%,j%
3200 PROCclear:PROCtitle("Delete Item",
1)
3210 I%=VAL(FNinput("Item number : ","
",0,4,4,48,57)):j%=I%
3220 IF I%=0 OR I%>N% PROCclear:ENDPROC
3230 PROCsee(I%):PROCcentre("Are you su
re ? (Y/N)",0):VDU23;11,0;0;0;0;0;0;
3240 REPEAT:K$=GET$:UNTIL INSTR("YyNn",
K$):IF K$="n" OR K$="N" PROCclear:ENDPRO
C
3250 f%=n%:b%=1%:PROCread(b%):n%=f%:PRO
Cwrite(b%):PROCread(f%):1%=b%:PROCwrite(
f%):IF S%=j% S%=f%
3260 IF j%<>N% PROCread(N%):f%=n%:b%=1%
:PROCwrite(j%):PROCread(b%):n%=j%:PROCwr
ite(b%):PROCread(f%):1%=j%:PROCwrite(f%)
:IF S%=N% S%=j%
3270 N%=N%-1:PROCclear:ENDPROC
3280 :
3290 REM ***** Save, Load & OSCLI *****
3300 :
3310 DEFPROCsave
3320 IF DC% AND ch%=0 PRINT'"No file op
en":TIME=0:REPEAT:UNTIL TIME>400:ENDPROC
3330 IF NOT(DC%) PROCclear:PROCtitle("S
ave File",1):PROCtitle("M-Base : "+N$,22
):N$=FNinput("Save file as :"+CHR$131,N$
,0,4,10,48,122)
3340 PROCstinfo
3350 IF NOT(DC%) PROCcentre("Saving "+N
$,6):OSCLI("SRSAVE "+N$+" 0 "+STR$~(E%))
```

```
:ELSE EXT#ch%=E%:CLOSE#ch%:ch%=0
 3360 PROCclear:ENDPROC
 3370 :
 3380 DEFPROCstinfo
 3390 E%=0:FOR A%=da% TO da%+&3FF STEP4:
!A%=0:NEXT
 3400 $da%=N$:!(da%+&C)=N%:!(da%+&10)=F%
:!(da%+&14)=L%:!(da%+&18)=S%:Q%=da%+&20
 3410 FOR A%=1 TO F%:$Q%=F$(A%):!(Q%+&10
)=L%(A%):!(Q%+&14)=X%(A%):!(Q%+&18)=Y%(A
%):Q%=Q%+&1C:NEXT
 3420 E%=&405+L%*N%:PROCio(FALSE,da%,da%
+&3FF,0)
 3430 ENDPROC
 3440 :
 3450 DEFPROCload
 3460 IF DC% o$="Open":ELSE o$="Load"
 3470 PROCclear:PROCtitle(o$+" File",1):
PROCtitle("M-Base",22):N$=FNinput("File
name :"+CHR$131,N$,0,4,10,33,122)
 3480 PROCcentre(o$+"ing "+N$,6):IF NOT(
DC%) OSCLI("SRLOAD "+N$+" 0"):ELSE ch%=O
PENUP N$
 3490 PROCassert:ENDPROC
 3500 :
 3510 DEFPROCassert
 3520 PROCio(TRUE,da%,da%+&3FF,0)
 3530 N$=$da%:N%=!(da%+&C):F%=!(da%+&10)
:L%=!(da%+&14):S%=!(da%+&18):Q%=da%+&20
 3540 FOR A%=1 TO F%:F$(A%)=$Q%:L%(A%)=!
(Q%+&10):X%(A%)=!(Q%+&14):Y%(A%)=!(Q%+&1
8):Q%=Q%+&1C:NEXT
 3550 PROCclear:ENDPROC
 3560 :
 3570 DEFPROCoscli:VDU23;11,255;0;0;0;0;
 3580 INPUT''"*"O$
 3590 IF O$="MX" OR O$="mx" PROCmset:END
PROC
 3600 ON ERROR GOTO3640
 3610 PROCclear:VDU23;11,0;0;0;0;:PROCti
tle("*"+O$,1):PROCtitle("MOS",22):VDU28,
0,20,39,4
 3620 OSCLI(O$):PRINT''"Press SPACE":VDU
26:ON ERROR VDU3:IF ERR=17 OSCLI("FX4"):
GOTO120:ELSE CLS:REPORT:PRINT" at line "
;ERL:END
 3630 REPEAT:UNTIL32=GET:ENDPROC
 3640 VDU26:PROCclear:PROCtitle("MOS err
or",1):PROCtitle(O$,22)
 3650 PRINTTAB(0,4);"MOS command error :
 ":REPORT
 3660 PRINT''"Press SPACE":ON ERROR VDU3
:IF ERR=17 OSCLI("FX4"):GOTO120:ELSE CLS
:REPORT:PRINT" at line ";ERL:END
 3670 REPEAT:UNTIL 32=GET
 3680 GOTO120
 3690 :
 3700 REM ***** View Information *****
 3710 :
 3720 DEFPROCinfo:PROCclear:VDU23;11,0;0
;0;0;0;
 3730 PROCtitle("Information",1):PROCtit
le("M-Base",22)
 3740 PRINTTAB(0,4);"File name :";CHR$13
1;N$
 3750 PRINTTAB(0,6);"No.Fields :";CHR$13
0;F%
 3760 PRINTTAB(0,8);"No. Items :";CHR$12
9;N%
 3770 IF F%=0 PROCcentre("Press SPACE",2
0):REPEAT:UNTIL INKEY-99:PROCclear:ENDPR
OC
 3780 PRINTTAB(0,10);:FOR A%=1 TO F% STE
P2:PRINTF$(A%);TAB(20);:IF A%<F% PRINTF$
(A%+1):IF F%<13 PRINT
 3790 NEXT:REPEAT:UNTIL INKEY-99:PROCcle
ar:ENDPROC
 3800 :
 3810 REM ***** Status & Setup *****
 3820 :
 3830 DEFPROCstatus:REPEAT
 3840 PROCclear:PROCtitle("Status",1):PR
OCtitle("M-Base",22)
 3850 PRINTTAB(0,4);"Options :"''" 1) Pr
inter       : ";:IF PR% PRINT"On":ELSE
PRINT"Off"
 3860 PRINT'" 2) Complex search : ";:IF
CS% PRINT"Any":ELSE PRINT"All"
 3870 IF F%>0 PRINT'" 3) Print options"
 3880 PRINT'" 4) Closeness list : ";MG%:
PRINT'" 5) CR - ";:IF CR% PRINT"on":ELSE
 PRINT"off"
 3890 PRINT'" 6) Media : ";:IF DC% PRINT
"Disc":ELSE PRINT"Sideways RAM"
 3900 PRINT'" 7) Main menu"
 3910 PRINT'"Press 1-7 : ";:o$="1234567"
:IF F%=0 o$="124567"
 3920 REPEAT:O$=GET$:UNTIL INSTR(o$,O$)
 3930 IF O$="1" PR%=NOT(PR%)
 3940 IF O$="2" CS%=NOT(CS%)
 3950 IF O$="3" PROCpopt
 3960 IF O$="4" INPUT'"New number "MG%:I
F MG%<1 OR MG%>50 MG%=15
 3970 IF O$="5" CR%=NOT(CR%)
```

```
 3980 IF O$="6" PROCswitch
 3990 UNTIL O$="7":ENDPROC
 4000 :
 4010 DEFPROCswitch
 4020 PRINT'CHR$(11);"Are you sure (Y/N)
 : ";:REPEAT:o$=GET$:UNTIL INSTR("YyNn",
o$):PRINTo$:IF o$="N" OR o$="n" ENDPROC
 4030 IF DC% AND ch%=0 DC%=FALSE:ENDPROC
 4040 PRINTCHR$(11);:IF DC% PRINT"Auto C
lose File         ":PROCsave:DC%=FALSE:PR
OCcleardb:ENDPROC
 4050 IF N$="" DC%=TRUE:ENDPROC
 4060 PRINT"Save file     (Y/N) :  ";CHR$
(8);:VDU7:REPEAT:o$=GET$:UNTIL INSTR("Yy
Nn",o$):PRINTo$
 4070 IF o$="Y" OR o$="y" PROCsave
 4080 DC%=TRUE:PROCcleardb:ENDPROC
 4090 :
 4100 DEFPROCpopt
 4110 FOR A%=1 TO F%:IF P%(A%) S$(A%)="O
n":ELSE S$(A%)="Off"
 4120 NEXT:PROCsee(0):PROCtitle("Printou
t Options",1):PROCtitle("Options",22)
 4130 v%=1:REPEAT
 4140 S$(v%)=FNinput(F$(v%),S$(v%),X%(v%
),Y%(v%),L%(v%),32,126)
 4150 IF INKEY-58 AND v%>1 v%=v%-1:ELSE
v%=v%+1
 4160 UNTIL v%>F%
 4170 FOR A%=1 TO F%:IF INSTR("OFFoffoff
",S$(A%)) P%(A%)=FALSE:ELSE P%(A%)=TRUE
 4180 NEXT:ENDPROC
 4190 :
 4200 DEFPROCmset
 4210 LOCAL c%,A%,A$
 4220 PROCclear:PROCtitle("M-Base SetUp"
,1):PROCtitle("File : MSetUp",22):PRINTT
AB(0,5);
 4230 c%=OPENIN"MSetUp":IF c%=0 PRINT"Fi
le not found":TIME=0:REPEAT:UNTIL TIME>=
700:PROCclear:ENDPROC
 4240 REPEAT:A$="":REPEAT:A%=BGET#c%:IF
A%>32 AND A%<127 A$=A$+CHR$(A%)
 4250 UNTIL A%=13 OR EOF#c%
 4260 IF FNbad(A$) PRINT"Bad line : ";A$
 4270 UNTIL EOF#c%:CLOSE#c%:TIME=0:REPEA
T:UNTIL TIME>200:PROCclear:ENDPROC
 4280 :
 4290 DEFFNbad(A$)
 4300 LOCAL f$,o%,p%,r$,c$
 4310 IF LEFT$(A$,1)<>"[" =TRUE
 4320 o%=INSTR(A$,"]"):IF o%=0 =TRUE
 4330 f$=MID$(A$,2,o%-2)
 4340 IF f$<>"*" AND FNlow(f$)<>FNlow(N$
) =FALSE
 4350 p%=INSTR(A$,":"):IF p%=0 =TRUE
 4360 c$=MID$(A$,o%+1,p%-o%-1):r$=RIGHT$
(A$,LEN(A$)-p%)
 4370 IF c$="PRINT" =FNsp(r$)
 4380 IF c$="CR" =FNcr(r$)
 4390 IF c$="LENGTH" =FNlength(r$)
 4400 IF c$="COMP" =FNcomp(r$)
 4410 IF c$="PRINTER" =FNprinter(r$)
 4420 =TRUE
 4430 :
 4440 DEFPROCcomm
 4450 INPUT''"[*]"C$:C$="[*]"+C$
 4460 IF FNbad(C$) PRINT'"Bad command :
Press SPACE";:REPEAT:UNTIL 32=GET
 4470 PROCclear:ENDPROC
 4480 :
 4490 DEFFNsp(r$)
 4500 IF LEN(r$)<F% =TRUE
 4510 FORA%=1 TO F%:P%(A%)=TRUE:IF MID$(
r$,A%,1)="0" P%(A%)=FALSE
 4520 NEXT:=FALSE
 4530 :
 4540 DEFFNcr(r$)
 4550 IF r$="ON" OR r$="on" OR r$="1" CR
%=TRUE:=FALSE
 4560 IF r$="OFF" OR r$="off" OR r$="0"
CR%=FALSE:=FALSE
 4570 =TRUE
 4580 :
 4590 DEFFNlength(r$)
 4600 LOCAL l%:l%=VAL(r$)
 4610 IF l%<3 OR l%>50 =TRUE
 4620 MG%=l%:=FALSE
 4630 :
 4640 DEFFNcomp(r$)
 4650 IF r$="all" OR r$="ALL" CS%=FALSE:
=FALSE
 4660 IF r$="any" OR r$="ANY" CS%=TRUE:=
FALSE
 4670 =TRUE
 4680 :
 4690 DEFFNprinter(r$)
 4700 IF r$="ON" OR r$="on" OR r$="1" PR
%=TRUE:=FALSE
 4710 IF r$="OFF" OR r$="off" OR r$="0"
PR%=TRUE:=FALSE
 4720 =TRUE
 4730 :
 4740 REM **** More to follow ****
```

BEEBUG Education

An unusual event: in this month's BEEBUG Education Mark Sealey looks at a new release for the 8-bit BBC micro.

Product	**Teddy Bear's Rainy Day**
Supplier	**Sherston Software** **Swan Barton, Sherston,** **Marlborough, Wilts SN16 0LH.** **Tel. 0666 840433,** **fax 0666 840048**
Price	**£26.95 ex. VAT**

TEDDY BEAR'S RAINY DAY

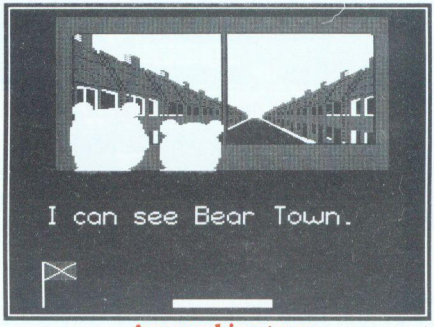
Good old Sherston! As the flow of new products for the BBC micro has all but dried up, one of the firms that has been on the scene since the beginning has brought out its last title for the BBC micro - and a good one, too.

Aimed at children in key stage 1 classes, Teddy Bear's Rainy Day is a topic-based package around the themes of bears, stories and the environment. Officially it is the sequel to the highly successful Teddy Bear's Picnic. But this farewell offering from Sherston can be used even if you are not familiar with the earlier product. At the start it should be said that everything you get if you decide to buy this pack (40 track DFS disc, teacher's and pupil's book, workcards and A3/A4 overlays) will delight you.

WHAT HAPPENS?

The package comes on two sides of the disc (a 3.5" ADFS version is available on request) and is booted in the usual way. An initial menu screen presents you with nine options. These are: read the instructions, teacher control, running side 2 and the five parts of the adventure.

Helpfully, the instruction screen takes the children through the various keys that are used in response to prompts throughout the program; these are subsequently used consistently in the software and make its use by even those most unused to the computer straightforward and easy.

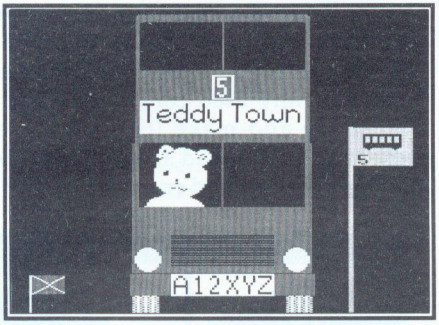


Approaching town

Teacher Control allows you to adjust the way the program operates... the number of players (one or two, only), the level of difficulty (of which there are three), sound control (again three levels), and enabling/disabling the concept keyboard. It is also possible to make, edit and select sequences of up to eight lists of children's names; these can be subsequently selected on this page so that groups can be called to the computer to tackle specific exercises in a predetermined order without any immediate intervention from the teacher or adult.

It is also possible to control some of these options by key presses. Shift-3, for

example, perhaps rather confusingly unless you can always remember it, toggles the player numbers. In all cases the options set are displayed where appropriate.



Crossing the road

Other keys control what is happening in the five games themselves. Again, these are by and large logical: at bath time, for example, 'P' toggles the plug being put in or out.

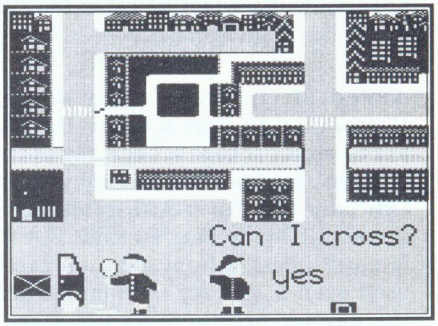
By the time that you have reached the end of the main menu and started the adventure you will have discovered how much thought has gone into this program: for example, children are not required to press the Shift key when entering their names. It capitalises whatever initial letter is entered automatically. During conversations, colours and screen location indicate the participating 'speaker'.

THE ADVENTURES THEMSELVES

There are five of these, which can either be tackled separately in any order or from the first (The Phone Call) to the last (Hot or Cold?).

The way that you actually approach these components will depend on how you intend to use the software as a whole. The fact that you have this choice is indicative of how far things have come in the design of this sort of pack. Flexibility of use has been emphasised at the expense of slick graphics and over-prescriptive and narrow "teaching points".

To decide on which parts of the adventure to use, how and in which order it is useful to read the introductory pages of the manual. In it, each adventure has a page to itself; what happens is briefly described, there is a graphic of what you can expect to see on screen, and details of the route through the scene. But, importantly, there is the author's assessment of the learning points that are covered.



Waiting for the bus

It is useful to know, for instance, that in The Phone Call, numbers entered by the children that are not valid in the UK are not allowed, or that the combinations of coins possible in the bus ride section, Fares Please, vary according to level: they are restricted to pennies in level one but allow any grouping at level three. This is the sort of information that makes planning any use of the program so easy.

The material covered in the five adventures centres around: letters and

numbers used in sequence (The Phone Call), geometry and safety awareness (A Walk in the Town), money and number facts (Fares Please), patterns and turtle geometry (Slide & Squelch), and opposites, temperature, timing and sequencing (Hot or Cold?).

There is a realistic progression through the events of the rainy day through which the teddy lives: he is invited out by Grandpa, travels to his house, gets mucky and has to clean himself up. It would thus be advisable for the children to experience the logic of this as a whole at least once before being expected to concentrate on particular aspects (geometry, number work or safety, for example) by use of the pre-set sequence facility that has been described earlier.

DOCUMENTATION

As has been said more than once before now, the booklet that comes with Teddy's Rainy Day is excellent - in common with all Sherston's material.

The book is crammed with sight vocabulary (the words that the children can both learn from the package and which they will need to help them with it). There are umpteen suggestions for use and brief background information, for example, on the invention of the telephone and the classic ways of making one with cups and string. There is an exhaustive booklist and some guidance on difficulties which you may meet. The workcards and overlays are colourful and the pack comes in a plastic wallet - all in all an addition to the software library to be proud of!

CONCLUSION

This review has gone into detail in describing and evaluating Teddy Bear's Rainy Day. This is because the product epitomises the state of the art after eleven years of educational software for the BBC micro. It is easy to use, still original, albeit simple in idea and execution, well supported by the documentation and ancillary resources supplied as part of the pack, amenable of use in a variety of teaching styles and attractively presented.

Teddy Bears Rainy Day

Above all, Teddy Bear's Rainy Day actually can almost be guaranteed to sponsor real learning: the language that will be used whilst children are 'playing' with Teddy, the opportunities for problem solving, the promotion of logical thinking. There are mathematical skills to be exercised, especially in number work and turtle geometry. And, perhaps most importantly, the chance for an imaginative teacher to draw on the product to engage the almost inevitably enthusiastic children in both art and discussions on personal health and safety issues.

Even now, you could do a lot worse than buy Teddy Bear's Rainy Day; it is sensibly priced and could still - thanks to its emphasis on child safety and well being - fill a gap in the curriculum.

BEEBUG WORKSHOP

Sorting Revisited

by David Peckett

Regular readers may remember that BEEBUG Vol.11 No.10 & Vol.12 No.1 looked at various ways of sorting data into order. Those articles prompted quite a lot of correspondence and so, this month's Workshop follows up some of the points raised by readers.

One result of the limited length allowed for BEEBUG Workshops is that sometimes, less information is included than one might wish. One such area in those articles was how to add an item into the right place in an already-sorted list. For instance, if you are keeping a list of club members, you will probably want to add new ones in alphabetical order.

One way is to re-sort the list as each item is added. To do this, though, you must choose the sorting algorithm very carefully so that the sort after each item is entered is very fast.

Here's where it comes in useful to know how the different sorts react to different data arrangements. Some are very much affected by the way that the data is already ordered, while others have pretty consistent run times. For example, the Shell sort is a good general-purpose sort, and normally much faster than the Bubble sort. In one particular case, however, the Bubble is noticeably faster. If the list is already in order, apart from only one item, then the correction can be made very quickly. This is because the sort only needs to make one pass through the list.

This is exactly the case we have when we add an item to a list. However, we do need a slightly more specialised program. The original Bubble sort started at the bottom (i.e. the low index numbers) of the array of data a n d bubbled items up to the top. If, however, we are adding data to the list, it is easier to add it at the top (e.g. by adding 1 to an array index); we therefore need to adapt the system to bubble (or should it be sink?) an item DOWN to the correct place.

Here is a revised version of the original PROCbubble to do the job. You will see that I have also changed it to access the data in array() via the pointers in ptr%() - this is the most likely need in database programs, etc.

SINKSORT

```
10000  DEF PROCsink(st%,fin%)
10010  IF st%>=fin% THEN ENDPROC
10020  LOCAL F%,I%
10030  FOR I%=st% TO fin%
10040    ptr%(I%)=I%
10050    NEXT
10060  REPEAT
10070    F%=FALSE
10080    FOR I%=fin% TO st%+1 STEP -1
```

```
10090       IF array(ptr%(I%-1))>
                 array(ptr%(I%))
              THEN PROCswap
10100       NEXT
10110     st%=st%+1
10120     UNTIL NOT F%
10130   ENDPROC

10500   DEF PROCswap
10510   LOCAL temp%
10520   temp%=ptr%(I%-1)
10530   ptr%(I%-1)=ptr%(I%)
10540   ptr%(I%)=temp%
10550   F%=TRUE
10560   ENDPROC
```

QUICKSORT

Several BEEBUG members referred to the 'Quicksort'. This is a very fast way of sorting long lists, although it's not so good for short ones. Unfortunately, it is a little tricky to explain.

Briefly, the sort finds the value in the middle of the array and then works in from each end to the middle. As it works in, the sort moves the individual elements around so that all in one half are greater than or equal to the middle value, while those in the other half are less than it. At that stage, it has done a very rough ordering.

Having done that, it splits the whole array into 2 halves, re-sorting each half into a high side and a low side. Each of the 2 halves is then split into 2 more halves and the procedure repeated. Then the quarters are halved... At the end of repeated halving, which goes as far as adjacent elements, the whole thing is found to be sorted. The technique is sometimes called 'Sorting from Both Ends'. What I hope that description shows is that the sort uses the same basic split and divide process on successively smaller parts of the array until it is finished. Such an approach is a perfect application for a recursive procedure.

All that recursion means is that a function is called by itself, which is called by itself..., doing a bit each time until a call creates an exit condition and the whole chain unravels. Here's some pseudo-code for an important recursive procedure:

```
1 DEF PROCDrink_Beer
2 Take a Swig
3 IF Glass NOT Empty PROCDrink_Beer
4 ENDPROC
```

Most versions of Basic make it almost impossible to use recursion, since all the variables used at each recursion level must be preserved while deeper calls are active. BBC Basic, though, supports recursion very well, with its FNs, PROCs and LOCAL variables. Here is a recursive Quicksort.

Line 11040 finds the middle value of *array()*, or the part of it sent to the procedure, and then the two REPEAT-UNTIL loops at lines 11060-11090 and 11100-11130 find misplaced elements either side of the centre point. Assuming that the 2 pointers have not crossed (why?) the elements are swapped.

QUICKSORT

```
11000   DEF PROCquicksort(st%,fin%)
11010   IF st%>fin% THEN ENDPROC
11020   LOCAL hiptr%,loptr%,midval
11030   loptr%=st%:hiptr%=fin%
11040   midval=array((st%+fin%)DIV 2)
11050   REPEAT
11060     loptr%=loptr%-1
11070     REPEAT
11080       loptr%=loptr%+1
11090       UNTIL array(loptr%)>=midval
11100     hiptr%=hiptr%+1
11110     REPEAT
11120       hiptr%=hiptr%-1
11130       UNTIL array(hiptr%)<=midval
11140     IF loptr%<=hiptr% THEN
            PROCqswap(loptr%,hiptr%)
11150     UNTIL loptr%>hiptr%
```

continued on page 24

Putting Directories To Work (Part 1)

Michael Shepherd makes ADFS easy.

Users of the ADFS welcome the advantages it has over the earlier DFS systems. To many, the main benefits are the longer filenames, making possible the use of more descriptive names for files, and the directory tree structure, enabling a greater degree of classification of files and a better use of disc storage space. However, there is an accompanying disadvantage, at least for the impatient, in that the longer filenames, and even longer pathnames, can mean a greatly increased amount of keyboard input before any particular file can be loaded.

One possible means of reducing this additional work is to convert the current directory into a menu format so that the choice of file or next directory can be made by means of a two-number input instead of the full LOAD/*DIR <name> command. As ADFS keeps a copy of the current directory in memory, it is possible to read this copy directly in order to form a suitable menu. On the Master 128, with the ADFS Version 1.50, this copy is stored on the five pages &C400 - &C8FF and from the information on directory structure given in the Reference Manual it is possible to extract all required information.

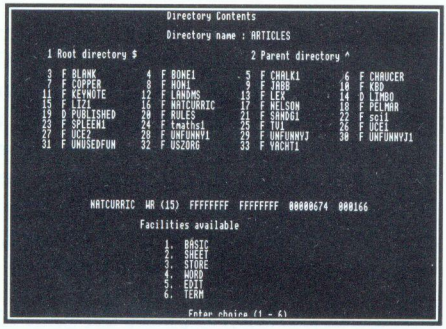
It is also of interest to note that the Free Space Map is copied to the two pages &C000 - &C1FF, the two intervening pages &C200 - &C3FF being used as a workspace and to store temporary information such as the pathname defined by a *LIB command. The structure of the ADFS directory is described in more detail in the technical notes below.

However, it is possible that in environments other than the M128, and even in other versions of the ADFS, the copy of the current directory will be stored elsewhere, so that a more generally applicable means of access to the current directory should be sought. This can be achieved by the use of OSWORD. OSWORD may be considered as an expansion of OSBYTE; it handles data in a similar manner, but is used to handle two or more bytes in the same operation - typically pages or sectors in a single call. In all, there are seven OSWORD calls, of which four are ADFS-only and three DFS-only. Parameters are passed to the call in a block of memory, to which *X* and *Y* act as pointers, rather than directly, as in OSBYTE calls.

Type in and save the *Contents* listing. It is assumed that, when in regular use, the program will be chained from the !BOOT file so that the starting point is to be the root directory ($). In addition, a function key (f9 in this case) is set to return the system to Basic, to the root directory and to re-run the program as and when required.

The ADFS-based program Contents copies the current directory from disc into user memory and from this composes a menu showing all possible choices in numbered sequence starting with '0' to end the program, '1' to return to the root directory and '2' to return to the parent directory, followed by a full listing of the directory contents accompanied by a 'D' or 'F' to indicate whether the entry is a directory or a file.

The program will only accept a numerical input in the valid range. When a valid choice is made, the program will identify it as a directory or a file. If a directory is selected, this is made the current directory and the process repeated. Otherwise, the choice is a file, and the user is invited to



'Contents' in action

choose a 'language' from those available, i.e. ROM based software like Basic, View etc. Line 1520 details the ROM calls, here based on fairly standard choices, you should change these to suit your own system. Note that the entries in this data line must be five characters long and contain the exact 'star' command that calls the ROM. The chosen file is included in a load command, stored on another function key, the language selected and the file loaded.

This part of the program could be developed further so that it takes you directly into an application with the file ready. Currently the loading of the file only works correctly with Basic, though any ROM can be called and files then loaded in the normal way. Some experimenting with *PROClang* should bring useful results.

TECHNICAL NOTES

Readers who do not have the Reference Manuals may wish to know more about the directory entries. All ADFS directories occupy five sectors of disc space, corresponding to five pages of memory, i.e. 1280 bytes, and the structure of a directory is as shown in Table 1. The first byte contains the Directory Master Sequence Number, which is the number appearing in brackets above the word 'Option' in the *CAT display and is the running total of all entries in the directory. The next four bytes are the string 'Hugo'. This is followed by the 47 blocks of 26 bytes giving the data for all the entries in the directory. The remainder of the directory holds its own name and access status, the start sector of the parent directory and the directory title. The directory ends with a &00, immediately preceded by a repetition of the string "Hugo".

Byte(s)	Content
000	Directory Master Sequence Number (BCD)
001-004	"Hugo"
005-01E	Details of first directory entry
01F-038	Details of second directory entry
(and in steps of 26 or 01A)	
4B1-4CA	Details of forty-seventh directory entry
4CB	0 (zero)
4CC-4D5	Directory name and access string
4D6-4D8	Start sector of parent directory
4D9-4EB	Directory title
4EC-4F9	Reserved
4FA	Directory Master Sequence Number (BCD)
4FB-4FE	"Hugo"
4FF	0 (zero)

Table 1. ADFS Directory structure.

An ADFS directory can contain up to 47 entries and the information relating to an entry is contained in 26 bytes as shown in Table 2. The first ten bytes, 0 - 9, contain the name of the entry, the first four bytes also containing the access status of the entry. If the entry name is less than 10

characters long, the name ends with a &0D (a carriage return). An entry starting with &00 indicates the end of the directory has been reached. The access attributes R,W,L,D are stored (in that order) by setting the 7th (most significant) bit of the appropriate byte. In this utility, only the D attribute is of interest and the value of the fourth byte MOD 128 is used to set the D or F label against the entry name.

Byte(s)	Content
00-09	Entry name and access string
0A-0D	Entry load address
0E-11	Entry execution address
12-15	Entry length in bytes
16-18	Entry start sector (on disc)
19	Entry sequence number

Table 2. Details of ADFS Directory entry

The next three sets of four bytes give respectively the load and execution addresses and length of the entry and are not used here. The next three bytes give the absolute start sector of the entry on the disc which is used by this program only if the entry is a directory. The final byte of the block contains the sequence number of the entry, which appears in both the *CAT and *INFO displays as the number in brackets against the entry name.

The technique used to record the access status explains the need to read the entry name byte-by-byte and to take the result MOD 128. Reading an entry name ceases after 10 bytes have been read, or a value of &0D is encountered. As stated previously, an entry name starting with &00 marks the end of the directory.

Setting the length of the pathname to its maximum permitted value of 255 characters means that a minimum of 24 levels of directory can be handled; the array element *pathlength%(x)* contains the total length of the pathname down to level x. Changes of directory are made by means of DIR <pathname> commands; *BACK and *DIR^ are avoided.

When the selected entry is not a directory but a file, the program places a LOAD command under a convenient function key (f5 in this case, but can be varied to suit the individual user) and then facilitates the choice of the required 'language'. A choice of six languages is given here, but the means for providing a wider or different choice is self-evident.

The call *FX138,0,133 places the value 133 in the keyboard buffer, where it remains until input is required - in this case after the program has ended. The value 133 is the input from function key 5 and implements any string stored there. In this case it is the LOAD command for the selected file. This call must be amended in line with any change to the function key used for the LOAD command.

In the next article I shall present a program which allows you to search a disc for a file name in any directory.

```
  10 REM Program Contents
  20 REM Version B1.0
  30 REM Author  J.M.Shepherd
  40 REM BEEBUG  December 1993
  50 REM Program subject to copyright
  60 :
 100 MODE 0:ON ERROR GOTO 1860
 110 DIM pathlength%(24):pathlength%(0)
=1:level%=0:H%=2
 120 pathname$=STRING$(255," "):pathnam
e$="$":OSCLI"DIR $"
 130 OSCLI"KEY9 *KEY5|M*BASIC|M*DIR$|MC
```

```
H."+CHR$34+"CONTENTS"+CHR$34+"|M"
  140 :
  150 REPEAT
  160 PROChugo(H%):PROCdisplay:PROCselec
t
  170 UNTIL ref%=-2
  180 CLS
  190 END
  200 :
 1000 DEF PROCdisplay
 1010 CLS:PRINTTAB(25);"Directory Conten
ts":M=&24CC:PROCstandardise(M)
 1020 PRINTTAB(25,2);"Directory name : "
;modified$
 1030 PRINTTAB(1,4);"1 Root directory $"
;TAB(41);"2 Parent directory ^"
 1040 FOR I=0 TO 44 STEP 4
 1050 FOR J=0 TO 3
 1060 M=&2005+(I+J)*&1A
 1070 IF ?M=0 I=44:J=3 ELSE PROCstandard
ise(M):PROCshow
 1080 NEXT
 1090 PRINT
 1100 NEXT
 1110 PRINTTAB(20,19);"Enter zero to exi
t program"
 1120 ENDPROC
 1130 :
 1140 DEF PROCselect
 1150 REPEAT
 1160 REPEAT
 1170 INPUTTAB(30,30);"Give number
";TAB(42,30);ref$
 1180 UNTIL LENref$<3 AND LENref$>0 AND
ASCref$<58 AND ASCref$>47 AND ASCRIGHT$(
ref$,1)<58 AND ASCRIGHT$(ref$,1)>47
 1190 ref%=VALref$-2
 1200 UNTIL ref%<(M-&1FEB)/&1A
 1210 IF ref%>0 AND ref%<48 M=ref%*&1A+&
1FEB:PROCstandardise(M)
 1220 IF ref%>0 ON (M?3 DIV 128)+1 PROCl
ang,PROCmove ELSE PROCmove
 1230 ENDPROC
 1240 :
 1250 DEF PROChugo(H%)
 1260 A%=&72:X%=&70:Y%=0:Z%=0
 1270 ?X%=0:X%!1=&2000:X%?5=8
 1280 X%?6=(H% AND &FF0000)/&10000
 1290 X%?7=(H% AND &FF00)/&100
 1300 X%?8=(H% AND &FF)
 1310 X%?9=5
 1320 X%?10=0
 1330 X%!11=0
 1340 CALL &FFF1
 1350 ENDPROC
 1360 :
 1370 DEF PROCshow
 1380 P=20*J:Q=6+I/4:C=70-2*(M?3 DIV 128
)
 1390 N$=RIGHT$(" "+STR$(I+J+3),2)
 1400 PRINTTAB(P,Q);N$;TAB(P+4,Q);CHR$C;
TAB(P+6,Q);modified$;
 1410 ENDPROC
 1420 :
 1430 DEF PROCstandardise(M)
 1440 modified$="":char%=0
 1450 REPEAT
 1460 A=?(M+char%) MOD &80
 1470 IF A<>13 modified$=modified$+CHR$A
:char%=char%+1
 1480 UNTIL A=13 OR char%=10
 1490 ENDPROC
 1500 :
 1510 DEF PROClang
 1520 N=0:data$="BASICSHEETSTOREWORD EDI
T TERM "
 1530 PRINTTAB(10,19);:OSCLI"INFO "+modi
fied$
 1540 PRINTTAB(20,21);"Facilities availa
ble";SPC(30)
 1550 FOR line%=1 TO 6
 1560 PRINTTAB(25,22+line%);STR$(line%);
".  "+MID$(data$,line%*5-4,5)
 1570 NEXT
 1580 PRINTTAB(30,30);"Enter choice (1 -
 6)";
```

```
 1590 REPEAT
 1600 lang%=GET-49:IF lang%<0 OR lang%>5
 VDU7
 1610 UNTIL lang%<6
 1620 lang$=MID$(data$,lang%*5+1,5)
 1630 IF lang%=2 OSCLI"BACK"
 1640 OSCLI"KEY5 LO."+CHR$34+modified$+C
HR$34+"|M*.|M|M*INFO "+modified$+"|M"
 1650 OSCLI"FX138,0,133":OSCLI lang$
 1660 ENDPROC
 1670 :
 1680 DEF PROCmove
 1690 IF ref%>0 PROCforwards ELSE PROCba
ckwards
 1700 OSCLI"DIR "+pathname$
 1710 ENDPROC
 1720 :
 1730 DEF PROCforwards
 1740 H%=(M!22 AND &FFFFFF):level%=level
%+1
 1750 pathlength%(level%)=pathlength%(le
vel%-1)+1+LENmodified$
 1760 pathname$=pathname$+"."+modified$
 1770 ENDPROC
 1780 :
 1790 DEF PROCbackwards
 1800 IF ref%=-1 level%=0:H%=2
 1810 IF ref%=0 H%=!&24D6 AND &FFFFFF:le
vel%=level%-1
 1820 pathname$=LEFT$(pathname$,pathleng
th%(level%))
 1830 ENDPROC
 1840 :
 1850 REM error handling
 1860 IF ERR=200 OSCLI"MOU.0":RUN
 1870 IF ERR DIV 16=10 CLS:PRINTTAB(20,1
5)"Hard break required"
 1880 REPORT:PRINT" at line ";ERL
```

BEEBUG Workshop (continued from page19)

```
11160  IF st%<hiptr% THEN
         PROCquicksort(st%,hiptr%)
11170  IF loptr%<fin% THEN
         PROCquicksort(loptr%,fin%)
11180  ENDPROC

12000 DEF PROCqswap(sptr1%,sptr2%)
12010 LOCAL temp
12020 temp=array(sptr1%)
12030 array(sptr1%)=array(sptr2%)
12040 array(sptr2%)=temp
12050 loptr%=loptr%+1
12060 hiptr%=hiptr%-1
12070 ENDPROC
```

This continues until the pointers HAVE crossed when, and only if there are elements to sort, the recursive calls to further sort the high and low halves are made. These, of course, repeat the process until only one element in *array()* has to be 'sorted', when that recursive chain unravels. Note that the procedure's parameters are LOCAL, as are the variables it uses. This means, for example, that *loptr%* in one level of the chain is different from the *loptr%* in the next level up or down - the system saves them all for us.

On the magazine disc, you will find a Quicksort demo program which shows very clearly how the method sorts in from each end. It also sorts the same items via a Shell sort, so you can see the speed difference of the two approaches.

Readers who would like to find out more about both searching and sorting are recommended to consult the book File Handling for All published by RISC Developments at £9.95 ex. VAT (zero rated at time of writing), p&p £2. An accompanying disc (specify 5.25" DFS or 3.5" ADFS) is also available for a further £2 to BEEBUG readers.

File Handling

Marshal Anderson looks at preserving your data.

Even the firstest of First Coursers will have used their disc drive for loading programs. However, you can get much more from your disc drive than that. If you use a word processor or database you will be familiar with the idea that data can be stored on disc. This is important for two reasons: first, it means that data dealt with by a program, be it a novel or just the high scores of a game, can be stored while the computer is switched off and then reloaded, into the original or another machine, at any time. Second, it allows the computer to deal with a lot more data within a program; the maximum RAM available in an Master 128 largely depends on the wit of the programmer, but you'll be lucky to get access to more than 32K at any one time. With a disc as part of the machine's memory, albeit a rather slow part, you might have access to nearly 700K. The question is - how is it done?

BBC Basic gives you a fairly straight forward set of commands to help you access the disc directly from inside your program, but first...

AN OVERVIEW

Data is stored on the disc sequentially in single bytes, each of which can be set to any number from 0 to 255. Text is stored in ASCII form, so a byte set to 65 would represent a capital 'A' and so on. The 'non-ASCII' bytes are used by Basic to tokenise key words; when you write files to the disc you can use any value for anything you want. A group of bytes saved together is called a *file*. Where exactly it goes on the disc is not our problem; DFS or ADFS sorts this out for us and keeps track of where everything is. If we want to do something with a file we first open it, then write and/or read our data, then we *close* it. Each file that we open has a *handle* by which we can identify it; by using handles we can access several files at once - up to five to be exact - though this is not recommended for your first foray into disc filing.

The rest of this article assumes you are using BBC Basic II or later - if you still have Basic I you should use OPENIN for both OPENIN and OPENUP.

So, the first thing we need to do is open a new file, for this we use OPENOUT, like this:

```
10 x=OPENOUT ("file")
```

Don't run this yet. The argument in brackets must be a string or a variable containing a string. In this case a file will be created on the disc called *file*, and the file handle will be stored in X. Let's put something in the file; the most straightforward way to do this is with PRINT#:

```
 20 PRINT#X, "Hello"
100 CLOSE#X
```

In line 20 the '#' tells Basic we want to print to a file and the X tells it which file to print to. Anything after the comma is then sent to the file. Finally we close the file with CLOSE#X.

AN IMPORTANT BIT

You *must* always keep track of the files you have opened and make sure they are

closed correctly, there are two reasons for this. The main one is that the output to the files is held in a buffer and the contents of the buffer are sent to the disc when the buffer is full. Each buffer is about 256 bytes long and, as your data is unlikely to divide exactly into this, there will be a bit left over. When you close a file the contents of the buffer are sent to the disc, whether the buffer is full is not. If you don't close it this final bit of data will be lost. The second reason is that Basic will not open the same file twice. If you try to open a file that is already open you will get the unsurprising error message 'Open'.

THE DREADED CLOSE#0

There is a quick and rather clumsy way of making sure that files are closed, and that is to use CLOSE#0. This will close all files that are open but it is not recommended as it seems to attach all sorts of rubbish to the end of the file. The extra is not necessarily a problem for your data, it is safe, it's just that it fills up a disc very quickly. Another reason for avoiding it is that, on all machines apart from the Compact, there is a bug which sometimes claims files are still open after you've closed them.

Let's get back to our program. OPENUP lets us read from the file as well as write to it, add the following lines to try this out:

```
30 CLOSE#X
40 X=OPENUP "file"
50 INPUT#X, A$
60 PRINT A$
```

INPUT#X reads back the information in the file and line 60 prints it on the screen.

This all seems very straightforward, *but*, it's important to remember that OPENOUT is for creating new files. If a file of that name already exists it will be deleted - there is no warning. If we want to read information in we must use OPENUP. This opens a file without deleting the information in it, but the file *must* exist on the disc so that it can be opened. Let's explore this, and some other ideas, in the more complex example shown at the end of this article. Type this program in and save it. It's a fairly straightforward task: get information on five employees and save them.

PROCgetinfo gets the employee data from the keyboard. Note the file types; *age* and *wage* must be numbers, *name$* and *id$* can be anything you like. *PROCprintinfo* prints the information, rather untidily, to the screen.

PROCcreatefile simply gets the name of the new file and opens, then closes it - it now exists on the disc. *PROCreadfromfile* is used if the file already exists. Here we use OPENUP. The file *must* exist or you will get an error. Note the order in which the variables are input from the file. This must be the same as the order in which they were printed to it. If you get this wrong you'll probably get a 'Type mismatch' error; you certainly won't get what you expect.

PROCwritetofile also uses OPENUP to write to an existing file. You must have entered a file name at options 1 or 2 for this to work. While you're using the PRINT# command this still effectively overwrites an old data in the file - you can't just change one entry - but in more sophisticated file handling this is possible, so using OPENUP is a good habit to get into. You'll also note that every procedure closes the file when it has finished.

Some of the errors you could come across while developing a file handling program are as follows. 'Open' means you have tried to open a file that is already open. 'Channel' means you have tried to access a file that does not exist. 'EOF' means end of file: you have tried to read beyond the end of the file. You also need to look out for things like 'Cat full' and 'Disc full', the normal limits of the disc still apply.

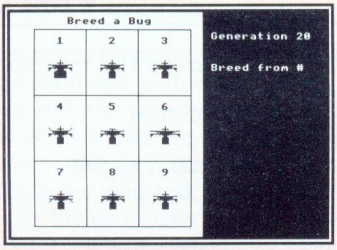
This article should have given you some ideas on how you can use the disc for saving data, it opens up all sorts of uses for your machine.

Next month I'll look at saving single bytes to disc - it's more useful than it sounds.

```
   10 REM Program Files
   20 REM Version B1.0
   30 REM Author  Marshal Anderson
   40 REM BEEBUG  December 1993
   50 REM Peogram subject to copyright
   60 :
  100 MODE 3
  110 PROCsetup
  120 REPEAT
  130 A=FNmenu
  140 IF A=1 PROCcreatefile
  150 IF A=2 PROCgetfilename:PROCreadfro
mfile
  160 IF A=3 PROCgetinfo
  170 IF A=4 PROCprintinfo
  180 IF A=5 PROCwritetofile
  190 UNTIL A>5
  200 END
  210 :
 1000 DEF FNmenu
 1010 PRINT "1) Create new file"'"2) Use
 existing file"'"3) Enter info"'"4) See
info"'"5) Save info"'"6) Finish"
 1020 INPUT Y
 1030 =Y
 1040 :
 1050 DEF PROCsetup
 1060 DIM NAME$(4),AGE(4),WAGE(4),ID$(4)
 1070 ENDPROC
 1080 :
 1090 DEF PROCcreatefile
 1100 INPUT"Name of file to create";FILE
$
 1110 X=OPENOUT FILE$
 1120 CLOSE#X
 1130 PRINT FILE$;" created"
 1140 ENDPROC
 1150 :
 1160 DEF PROCgetfilename
 1170 INPUT"What is the name of your fil
e";FILE$
 1180 ENDPROC
 1190 :
 1200 DEF PROCwritetofile
 1210 X=OPENUP FILE$
 1220 FOR Y=0 TO 4
 1230 PRINT#X, NAME$(Y),AGE(Y),WAGE(Y),I
D$(Y)
 1240 NEXT Y
 1250 CLOSE#X
 1260 ENDPROC
 1270 :
 1280 DEF PROCreadfromfile
 1290 X=OPENUP FILE$
 1300 FOR Y=0 TO 4
 1310 INPUT#X, NAME$(Y),AGE(Y),WAGE(Y),I
D$(Y)
 1320 NEXT Y
 1330 CLOSE#X
 1340 ENDPROC
 1350 :
 1360 DEF PROCgetinfo
 1370 CLS
 1380 PRINT"You are going to enter detai
ls for 5 employees. You need name, age,
wage and ID code."
 1390 FOR Y=0 TO 4
 1400 PRINT "Number ";Y
 1410 INPUT "Name ";NAME$(Y)
 1420 INPUT "Age ";AGE(Y)
 1430 INPUT "Wage FALSE";WAGE(Y)
 1440 INPUT "ID code";ID$(Y)
 1450 NEXT Y
 1460 PRINT "Finished"
 1470 ENDPROC
 1480 :
 1490 DEF PROCprintinfo
 1500 CLS
 1510 FOR Y=0 TO 4
 1520 PRINT NAME$(Y),AGE(Y),"       FALS
E";WAGE(Y),ID$(Y)
 1530 NEXT Y
 1540 ENDPROC
```

Best of BEEBUG

Applications I Disc

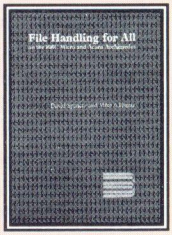


BUSINESS GRAPHICS - for producing graphs, charts and diagrams
VIDEO CATALOGUER - catalogue and print labels for your video cassettes
PHONE BOOK - an on-screen telephone book which can be easily edited and updated
PERSONALISED LETTER-HEADINGS - design a stylish logo for your letter heads
APPOINTMENTS DIARY - a computerised appointments diary
MAPPING THE BRITISH ISLES - draw a map of the British Isles at any size
SELECTIVE BREEDING - a superb graphical display of selective breeding of insects
THE EARTH FROM SPACE - draw a picture of the Earth as seen from any point in space
PERSONALISED ADDRESS BOOK - on-screen address and phone book
PAGE DESIGNER - a page-making package for Epson compatible printers
WORLD BY NIGHT AND DAY - a display of the world showing night and day for any time and date of the year

File Handling for All

on the BBC Micro and Acorn Archimedes

by David Spencer and Mike Williams



Computers are often used for file handling applications yet this is a subject which computer users find difficult when it comes to developing their own programs. *File Handling for All* aims to change that by providing an extensive and comprehensive introduction to the writing of file handling programs with particular reference to Basic.

File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File Handling and Databases on the Beeb and the Arc. However, all the file handling concepts discussed are relevant to most computer systems, making this a suitable introduction to file handling for all.

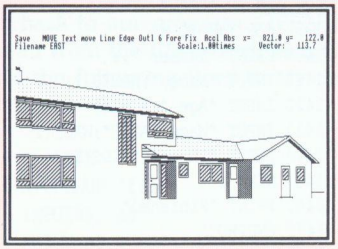
The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical and relational database design, and the book concludes with a chapter of practical advice on how best to develop file handling programs.

The topics covered by the book include:

Card Index Files, Serial Files, File Headers, Disc and Record Buffering, Using Pointers, Indexing Files, Searching Techniques, Hashing Functions, Sorting Methods, Testing and Debugging, Networking Conflicts, File System Calls

The associated disc contains complete working programs based on the routines described in the book and a copy of Filer, a full-feature Database program originally published in BEEBUG magazine.

ASTAAD



Enhanced ASTAAD CAD program for the Master, offering the following features:

- full mouse and joystick control
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

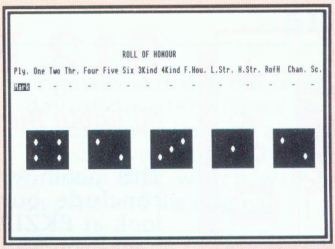
	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£5.95	**ASTAAD** (3.5" ADFS)	1408a	£5.95
Applications II (80 track DFS)	1411a	£4.00	**Applications II** (3.5" ADFS)	1412a	£4.00
Applications I Disc (40/80T DFS)	1404a	£4.00	**Applications I Disc** (3.5" ADFS)	1409a	£4.00
General Utilities Disc (40/80T DFS)	1405a	£4.00	**General Utilities Disc** (3.5" ADFS)	1413a	£4.00
Arcade Games (40/80 track DFS)	PAG1a	£5.95	**Arcade Games** (3.5" ADFS)	PAG2a	£5.95
Board Games (40/80 track DFS)	PBG1a	£5.95	**Board Games** (3.5" ADFS)	PBG2a	£5.95

All prices include VAT where appropriate. For p&p see Membership page.

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS.

Best of BEEBUG

Board Games



Solitaire - an elegant implementation of this ancient and fascinating one-player game, and a complete solution for those who are unable to find it for themselves.

Roll of Honour - Score as many points as possible by throwing the five dice in this on-screen version of 'Yahtze'.

Patience - a very addictive version of one of the oldest and most popular games of Patience.

Elevenses - another popular version of Patience - lay down cards on the table in three by three grid and start turning them over until they add up to eleven.

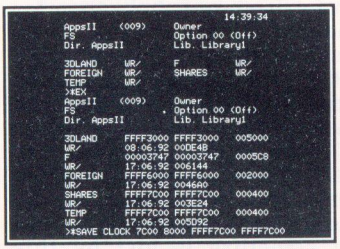
Cribbage - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

Twiddle - a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern.

Chinese Chequers - a traditional board game for two players, where the object is to move your counters, following a pattern, and occupy the opponent's field.

Aces High - another addictive game of Patience, where the object is to remove the cards from the table and finish with the aces at the head of each column.

Applications II Disc



Crossword Editor - for designing, editing and solving crosswords

Monthly Desk Diary - a month-to-view calendar which can also be printed

3D Landscapes - generates three dimensional landscapes

Real Time Clock - a real time digital alarm clock displayed on the screen

Running Four Temperatures - calibrates and plots up to four temperatures

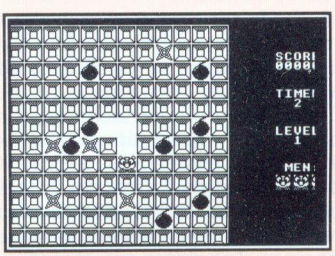
Julia Sets - fascinating extensions of the Mandelbrot set

Foreign Language Tester - foreign character definer and language tester

Share Investor - assists decision making when buying and selling shares

Label Processor - for designing and printing labels on Epson compatible printers

Arcade Games



George and the Dragon - Rescue 'Hideous Hilda' from the flames of the dragon, but beware the flying arrows and the moving holes on the floor.

Ebony Castle - You, the leader of a secret band, have been captured and thrown in the dungeons of the infamous Ebony Castle. Can you escape back to the countryside, fighting off the deadly spiders on the way and collecting the keys necessary to unlock the coloured doors?

Knight Quest - You are a Knight on a quest to find the lost crown, hidden deep in the ruins of a weird castle inhabited by dangerous monsters and protected by a greedy guardian.

Pitfall Pete - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way.

Builder Bob - Bob is trapped on the bottom of a building that's being demolished. Can you help him build his way out?

Minefield - Find your way through this grid and try to defuse the mines before they explode, but beware the monsters which increasingly hinder your progress.

Manic Mechanic - Try to collect all the spanners and reach the broken-down generator, before the factory freezes up.

Quad - You will have hours of entertainment trying to get all these different shapes to fit.

	Stock Code	Price		Stock Code	Price
File Handling for All Book	BK02b	£ 9.95			
File Handling for All Disc (40/80T DFS)	BK05a	£ 4.75	**File Handling for All** Disc (3.5" ADFS)	BK07a	£ 4.75
Joint Offer book and disc (40/80T DFS)	BK04b	**£ 11.95**	**Joint Offer book and disc** (3.5" ADFS)	BK06b	**£ 11.95**
Magscan (40 DFS)	0005a	£ 9.95	**Magscan Upgrade** (40 DFS)	0011a	£ 4.75
Magscan (80T DFS)	0006a	£ 9.95	**Magscan Upgrade** (80T DFS)	0010a	£ 4.75
Magscan (3.5" ADFS)	1457a	£ 9.95	**Magscan Upgrade** (3.5" ADFS)	1458a	£ 4.75

All prices include VAT where appropriate. For p&p see Membership page.

Tel. (0727) 840303 Fax. (0727) 860263

512 Forum

by Robin Burton

This month we conclude our look at PKZIP back-ups and see how PKUNZIP reverses the archiving process when disaster strikes.

Unfortunately space won't allow us to investigate all the PKZIP options, but we should be able to cover enough of them to allow you to build a reasonable back-up system to cater for most needs.

RECURSING DIRECTORIES

So far we've talked only about securing single directories, even though the batch job to do it may be one of a number doing different back-ups but which are related. One reason for that approach is that it makes PKZIP basics easier to explain, but equally very often when you update files they're all in one directory, so that's frequently how you'd use it anyway.

However, there's a trade-off between keeping each back-up job separate for simplicity and ending up with lots of jobs that are each too small to be practical. Obviously on a winchester the directory structure can be more complex than on floppies, though for efficient performance it's a good idea not to get too carried away with this idea. But in general, even on a hard drive, the more directories you have the smaller they tend to be. Conversely, on floppies despite the smaller capacity there's often a need for a directory structure, whether it's demanded by an application or your own convenience.

There comes a point therefore, where single directory back-ups, whether from a floppy or from a hard disc, would require several batch jobs but the amount of data just doesn't justify the complexity. The answer in these cases is directory recursion.

As the name implies, this option tells PKZIP to base the start of its activities on the specified or current directory as before, but also to include sub-directories in the operation. Like many PKZIP options the letter used for the switch is logical, 'r' (recurse) but this option must be used carefully or the results may not be quite what you expect.

POINTS TO WATCH

Judging by my mail, difficulty arises in recursion for back-ups more than for most other PKZIP operations. To be blunt this is a case of not reading the documentation thoroughly, but even so it's quite common. Although a command in the general form:

```
PKZIP -r <archive-name>
```

will certainly recurse all subdirectories in the current path and compress all the files found too, as it stands the original directory structure is lost by this command. This means that if you subsequently recover files from an archive created by the above command, although the back-up will have been completed OK, all the files will be extracted into a single directory.

That's not to say that recursion used this way isn't useful, it just isn't suitable for back-ups. An illustration will make this clearer, so let's take a look.

Suppose you have a directory which contains two or three subdirectories, the contents of each of which is similar but they each include several different file-types. Further, if some of the files are temporary working files which needn't be secured (say with the extension .WRK) while all the permanent files have varying extensions, then the job of extracting the permanent files from several subdirectories by normal DOS methods, assuming you want to keep the temporary files, would be awkward to say the least.

However, recursion (with exclusion) offers a very easy way to archive all these permanent files in one simple operation. Refer to last month's article for the exclusion option if you need a refresh.

Assuming the PKZIP command were issued from the top-level directory of this structure, to compress all the permanent files into a single archive file the command would be:

```
PKZIP -r <archive-name> -x *.wrk
```

This would recurse all subdirectories in the current directory, compressing all files into the specified archive file except those excluded by the '-x', in other words those with a .WRK extension. As in my root directory back-up, the exclusion directive can also be in the form '-X@' followed by a filename containing a list of excluded files. In this case any number of files could be excluded by the list, using wildcarded names, by explicitly naming each file in turn or by any mixture of the two.

Likewise, to back-up all the files of one particular type from a directory structure is just as easy if not easier. Suppose that we wanted just files with a .TXT extension, then the command:

```
PKZIP -r <archive-name> *.txt
```

is all that's needed. Nothing else but files with a .TXT extension will be archived by this command. Obviously you can also specify other options to refine this operation, including those described last month, or a number of others which could specify files older (or newer) than a given date, those not already archived and so on.

Used like this, PKZIP recursion offers powerful facilities which can make some jobs much easier, but of course our main interest here is security back-ups, so let's get back to that.

RETAINING STRUCTURE

What's usually needed for back-ups is an archive file which safeguards not only the files it contains, but also retains full information on how the files were originally stored. If this is done, when the day of disaster arrives such as a FAT corruption or the kids using your disc as a frisbee you can easily re-create a new master disc which will be an exact duplicate of the original.

The 'p' option (path) is used to preserve the full path structure of subdirectories, but note that this is another PKZIP directive which is case sensitive, the lower case version is what we need here (upper case 'P' recurses only specified directories.) When the command is intended to not only recurse sub-directories but also to preserve the entire directory structure, the command therefore is:

```
PKZIP -rp <archive-name>
```

This tells PKZIP to recurse subdirectories (from the current or specified directory) but also to record the original path of each of the files being archived.

Archives created like this can then be recovered in their entirety, which we'll examine next, to totally re-create a lost disc or path, but despite that they lose no flexibility over simpler archives. These can also be searched to recover a single file like a simple archive, and they can also be updated selectively by any of the methods we've previously looked at.

When you have several small directories sharing the same path and want to secure not only the data but the structure too, this is how it's done. I tend not to use recursion very much because on a hard disc most paths contain too much data for a single back-up floppy even when the data is compressed, but for floppy users, or if directories are small, recursion can simplify (and shorten) the job considerably.

RECOVERING

So far we've looked at several of the most useful basic PKZIP options for backing up files and directories, but having given you the essentials I'll leave you to explore the less commonly used archiving options yourself.

It's now time to examine the reverse process. Files in a .ZIP archive can't be accessed directly, though there are tools such as SHEZ which can help you to view the contents of a .ZIP file 'in situ'. However, I tend not to use such programs; again I choose what I think is the the simplest route, PKUNZIP.

We saw the simplest PKUNZIP command last month:

```
PKUNZIP -t <archive-name>
```

This tells PKUNZIP to test the integrity of every file in the specified archive. Each filename is displayed in turn, including its original pathname if appropriate, and PKUNZIP then checks the file against its 32-bit CRC. If all is well it says 'OK' for each file, while any discrepancy causes a report that the file is damaged. If you have the complete ZIP package you'll have a program called PKZIPFIX, which can in some circumstances recover individual files within an archive, but which will in any case fix the .ZIP file so that its other undamaged contents can still be recovered. Of course, most of the time there's no problem, so mainly I treat this option as an easy way of checking what a particular archive contains.

In my own experience the only reason for problems is the back-up floppy itself, so the best insurance against that sort of trouble is duplicated back-ups, preferably not kept in the same place. I can tell you that only once have I ever had two faulty back-ups of the same data, but in that case both discs had developed a bad sector, so PKZIP wasn't responsible.

A second potential cause of trouble is your version of the PKZIP software. To extract files from a .ZIP file obviously requires PKUNZIP, which is of course supplied as part of the PKZIP package. However, PKUNZIP is often supplied on its own on shareware discs to allow you to unpack the shareware disc's contents. The point to watch is that the version of PKUNZIP used is at least as recent as the version of PKZIP that produced the archive file.

The copy of PKUNZIP on a shareware disc will certainly handle everything on that particular disc, but do make sure your master copy of ZIP and UNZIP is the latest in your possession and that they're both the same version.

If you try to use a version of UNZIP to extract from a .ZIP file produced by a later version of PKZIP you'll probably get the message "Sorry, I don't know how to handle this file". This isn't inevitable, all later PKZIP versions do support all previous compression standards, and an old form of compression could have been specified during compression (this is another option) but if you do see this message there's no alternative, you need a later version on PKUNZIP.

PKUNZIP is at least as easy to use as PKZIP and the general format of commands is similar too, as you'd expect. For example, the command to extract all the files from an archive called 'SECURITY' on drive A: to the current directory on the current drive would be:

```
PKUNZIP a:security
```

As before the option is defaulted, so it becomes '-e' for 'extract', the .ZIP extension on the archive file is defaulted too, and the destination for the output data is the current directory and drive since no output path is supplied. Of course an output path can be specified too, so for example:

```
PKUNZIP a:security c:\recoveries\*.*
```

will unzip all the files in the archive to a directory called \RECOVERIES on drive C:, but note that in this case the target directory must pre-exist. Naturally files are most often restored to their original location, so in practice this option isn't needed very often.

Remember that if the archive contains only files and no directory structure data, PKUNZIP won't know anything about where they came from, so recovery is always to the current directory unless you specify otherwise. Of course, if the

archive included recursed directories with paths preserved (PKZIP options -rp), PKUNZIP will still allow you to recover any file to any directory, but life will be much easier for full recoveries.

For example, if a master disc is a total write-off or when you've had to re-partition a hard disc, the original directories won't exist, so you have two options.

One is to create the required directories yourself before the recovery, perhaps in your recovery batch files, but the other and easier option is to let PKUNZIP do it for you completely automatically. If the data was secured with both the 'r' and 'p' directives the original path is preserved along with the files, so:

```
PKUNZIP -d <archive-name>
```

can be used to tell PKUNZIP to recover all the archived files to their original directories. If the target directories don't exist at the time, PKUNZIP will create them for you as it goes along; if they do already exist they're used.

At this point I have a small confession to make. I deliberately omitted mention of the (specified) path and recursion options from my back-up examples for clarity, but in my live system I always use them, even if I know there's only one directory to secure. This is so that, if I have to perform full recovery, PKUNZIP will rebuild the complete subdirectory structure of my hard disc for me, while if I need to recover only one or two files I can still do that into any directory I want, including the original location if necessary.

To make sure this point is clear, consider the ES archive routines again and mentally add this extra information. All the \ES subdirectory back-ups take place from within \ES itself, so for \ES\SOURCES for example, the command to back-up to drive A: is:

```
PKZIP -rP a:es_srces \es\sources\*.*
```

This retains the full path of \ES\SOURCES in the archive and secures all its files too. When I recover, if either \ES or \ES\SOURCES doesn't exist they will be created for me; if they do exist it doesn't affect the recovery.

Note that I use the upper case path option, otherwise all the other \ES subdirectories would be included too and the job would fail because the back-up disc would become full. Also note a couple of other important items. Because I've used the specified path option I have to add the full path (from the root) of the directory to be archived, and I have to specify at the end of the path that all files (*.*) are to be included. Without the wildcarded filename only the directory structure of \ES\SOURCES would be saved and if SOURCES did contain any subdirectories these too would be ignored.

Although PKZIP operations can be extremely simple, when you use some of the more sophisticated options be aware that things do get complicated and your file and directory specifications must be absolutely precise. If you do need such options I'd advise that you test commands very thoroughly manually and ensure that the secured data can be accessed in exactly the way you expect too. Only when you're absolutely sure everything works should you build these commands into batch routines on which the safety of your data will depend. Remember, if eventually you need to recover data these routines MUST work; it will be too late by then if you find they don't.

AND FINALLY

That rounds off our look at PKZIP, but watch this space for other (I hope) interesting disc offers.

At the moment next month's topic is as much of a mystery to me as it is to you but in the meantime if anyone has specific queries on the points covered in the last three articles drop me a line and I'll try to help. If you do write, on this or any other topic, please note that I have recently changed jobs and in consequence replies to your letters may take a little longer in future.

```
01,5000):REM 2 Spaces
 2300 REPEAT
 2310 PROCsetpointvar:PROCpointer
 2320 PROCprintpos("Press 'Q' to quit.
"):REM 2 Spaces
 2330 PROCcurkey:*FX21,0
 2340 IF NOT end% THEN PROCshape("Filled
 Rectangles.  ",101,0)
 2350 UNTIL end%
 2360 end%=FALSE
 2370 ENDPROC
 2380 :
 2390 DEF PROCcomplexshape(name$,code%,d
ely%)
 2400 PRINTTAB(0,0)name$:PROCwait(dely%)
 2410 REPEAT
 2420 PROCsetpointvar
 2430 PROCprintpos("RETURN to set P. one
")
 2440 PROCpointer:PROCcurkey:*FX21,0
 2450 IF return% THEN x1%=x%:y1%=y%
 2460 UNTIL return%
 2470 return%=FALSE
 2480 REPEAT
 2490 x2%=x%:y2%=y%
 2500 PROCsetpointvar
 2510 PROCprintpos("RETURN to fix P. two
")
 2520 PROCpointer:MOVE x1%,y1%:MOVE x2%,
y2%
 2530 PLOT(code%+1),x1%,y1%
 2540 PROCcurkey:*FX21,0
 2550 MOVE x1%,y1%:MOVEx2%,y2%
 2560 PLOT(code%+1),x1%,y1%
 2570 UNTIL return%
 2580 return%=FALSE
 2590 REPEAT
 2600 x3%=x%:y3%=y%
 2610 PROCsetpointvar
 2620 PROCprintpos("RETURN to fix P. 3
"):REM 2 spaces
 2630 PROCpointer:MOVE x1%,y1%:MOVEx2%,y
2%
 2640 PLOT(code%+1),x3%,y3%:PROCcurkey:*
FX21,0
 2650 MOVE x1%,y1%:MOVEx2%,y2%
 2660 PLOT(code%+1),x3%,y3%
 2670 UNTIL return%
 2680 GCOLpattern%,col%
 2690 MOVE x1%,y1%:MOVE x2%,y2%
 2700 IF draw% PLOTcode%,x3%,y3%
 2710 return%=FALSE
 2720 ENDPROC
 2730 :
 2740 DEF PROCellipse
 2750 PRINTTAB(0,0) SPC 40
 2760 REPEAT
 2770 PROCprintpos("Press 'Q' to quit.
"):REM 2 Spaces
 2780 PRINTTAB(0,0)"(F)ill or (O)utline?
"
 2790 PROCsetpointvar:PROCpointer:PROCcu
rkey:*FX21,0
 2800 IF keypress% = 70 OR keypress% = 1
02 THEN PROCcomplexshape("Filled Ellipse
s.     ",205,5000)
 2810 IF keypress% = 79 OR keypress% = 1
11 THEN PROCcomplexshape("Outline Ellips
es.    ",197,5000)
 2820 UNTIL end%
 2830 end%=FALSE
 2840 ENDPROC
 2850 :
 2860 DEF PROCpar
 2870 PRINTTAB(0,0) SPC 40
 2880 PROCcomplexshape("Parallelograms.
    ",117,5000):REM 5 spaces
 2890 REPEAT
 2900 PROCsetpointvar:PROCpointer
 2910 PROCprintpos("Press 'Q' to quit.
"):REM 2 spaces
 2920 PROCcurkey:*FX21,0
 2930 IF NOT end% THEN PROCcomplexshape(
"Parallelograms.     ",117,0):REM 5 Spac
es
 2940 UNTIL end%
 2950 end%=FALSE
 2960 ENDPROC
 2970 :
 2980 DEF PROCarc
 2990 PRINTTAB(0,0) SPC 40
 3000 PROCcomplexshape("Arcs.
    ",165,5000):REM 15 Spaces
 3010 REPEAT
 3020 PROCsetpointvar:PROCpointer
 3030 PROCprintpos("Press 'Q' to quit.
")
 3040 PROCcurkey:*FX21,0
 3050 IF NOT end% THEN PROCcomplexshape(
"Arcs.                ",165,0):REM 15 Spa
```

```
ces
 3060 UNTIL end%
 3070 end%=FALSE
 3080 ENDPROC
 3090 :
 3100 DEF PROCsegment
 3110 PRINTTAB(0,0) SPC 40
 3120 PROCcomplexshape("Segments.
    ",173,5000):REM 11 Spaces
 3130 REPEAT
 3140 PROCsetpointvar:PROCpointer
 3150 PROCprintpos("Press 'Q' to quit.
")
 3160 PROCcurkey:*FX21,0
 3170 IF NOT end% THEN PROCcomplexshape(
"Segments.          ",173,0):REM 11 Spa
ces
 3180 UNTIL end%
 3190 end%=FALSE
 3200 ENDPROC
 3210 :
 3220 DEF PROCtext
 3230 PRINTTAB(0,0) SPC 40
 3240 INPUTTAB(0,0)"Text?
"text$:REM 15 Spaces
 3250 INPUTTAB(0,0)"X pos?
"xpos%:REM 14 Spaces
 3260 IF xpos% < 1 THEN xpos% = 1
 3270 REPEAT
 3280 IF xpos% + LEN text$ > 19 THEN xpo
s% = xpos% - 1
 3290 UNTIL xpos% + LEN text$ <= 19
 3300 INPUTTAB(0,0)"Y pos?
"ypos%:REM 14 Spaces
 3310 IF ypos% < 3 THEN ypos% = 3
 3320 IF ypos% > 30 THEN ypos% = 30
 3330 PRINTTAB(0,0) SPC 40
 3340 PRINTTAB(xpos%,ypos%)text$
 3350 PROCborder:MOVE x%,y%
 3360 ENDPROC
 3370 :
 3380 DEF PROCsetpointvar
 3390 xcone% = x%-50:ycone% = y%-50
 3400 xctwo% = x%-25:yctwo% = y%-75
 3410 ENDPROC
 3420 :
 3430 DEF PROCprintpos(slstring$)
 3440 IF draw% AND col%<>8 THEN COLOUR c
ol% ELSE COLOUR 7
 3450 PRINTTAB(0,0)"Step";step%;"   ":RE
M 3 Spaces
 3460 PRINTTAB(8,0)"X=";x%;"    ":REM 4
Spaces
 3470 PRINTTAB(15,0)"Y=";y%;"  ":REM 2 S
paces
 3480 PRINTTAB(0,1)slstring$
 3490 ENDPROC
 3500 :
 3510 DEF PROCpointer
 3520 MOVExcone%,ycone%
 3530 MOVExctwo%,yctwo%
 3540 PLOT86,x%,y%:MOVEx%,y%
 3550 ENDPROC
 3560 :
 3570 DEF PROCwait(waittime%)
 3580 FOR wait% = 1 TO waittime%
 3590 NEXT wait%
 3600 ENDPROC
 3610 :
 3620 DEF PROCloader
 3630 load% = FALSE
 3640 REM *MOUNT ADFS ONLY
 3650 PRINTTAB(0,0) SPC 40
 3660 INPUTTAB(0,0)"Filename?"file$
 3670 OSCLI ("LOAD ")+file$+(" 3000")
 3680 PRINTTAB(0,0) SPC 20
 3690 PROCinit
 3700 ENDPROC
 3710 :
 3720 DEFPROCsaver
 3730 save% = FALSE
 3740 REM *MOUNT ADFS ONLY
 3750 PRINTTAB(0,0) SPC 40
 3760 INPUTTAB(0,0)"Filename?"file$
 3770 OSCLI ("SAVE")+file$+(" 3000 8000"
)
 3780 PRINTTAB(0,0) SPC 20
 3790 ENDPROC
 3800 :
 3810 DEFPROCborder
 3820 GCOL0,1
 3830 MOVE 0,0:DRAW 0,944:DRAW 1279,944:
DRAW 1279,0:DRAW 0,0
 3840 GCOL 0,2
 3850 MOVE 0,954:DRAW 1279,954:DRAW 1279
,1024:MOVE 0,1024:DRAW 0,954
 3860 ENDPROC
```

Machine Code Corner

Mr Toad shows you how to call a Basic program with a festive star.

Happy Wossname, Toad-fans, and the Condiments of the Seasoning to you. Right, that's the humbug over - now for some real jollities.

Bill Woodall of Yeovil asked for me for help with the Watford Shadow RAM board fitted to his steam-powered model B; he wants some routines to shift blocks of data in and out of the shadow RAM, in connection with a database. Alas, I can't help him - I don't have either piece of hardware available and can't see what's wrong with the listing he sent me. The beastie seems to use addresses in main RAM as 'ports', not the SHEILA addresses I'm used to. So if you have one of these boards, please let me know and I'll put you in touch with Bill.

Arthur Adams wanted to know if it is possible to tack a Basic program onto a Sideways ROM header and call it with a star command. I knew it was, because the software with my EPROM blower does just that, but the programming is totally primitive. So, I thought I'd have a go for myself because I still have lots of space on the Toad ROM 90. I could tack on my Basic program that sets the Master's clock, calling it with a short star command instead of having to hunt for the disc. Readers without EPROM blowers might not have so much use for such a hybrid, but you never know, and the programming turned out to be quite interesting.

The method is obvious. It is simply impossible to get Basic to run in Sideways RAM, but what you can do is write a routine which copies a Basic program from sideways memory back into its normal habitat at PAGE, and then gets it going by some equivalent of RUN. You'd *SRWRITE the Basic program into the SRAM slot after the machine code, and *SRSAVE the whole lot to disc. A cheap 'n' cheerful way of creating a new star command - *but*, unlike a 'proper' star command, this will overwrite anything in the lower part of main memory, such as a program or document you may be working on.

I took my small Sideways ROM header (Beebug Vol.11 No.7), altered *.starCommand* in line 1430 from ABC to CK for 'CLOCK' - you can use any command you like. I tidied up by altering the *HELP text, title and copyright strings, getting rid of the test line 1460 and the unwanted definitions of OSWRCH, OSNEWL and OSWORD. All this takes longer to explain than to do.

Now we need two pairs of page zero addresses to act as indexes for the copying. Normally, 0 and 1 point to TOP, the top of the Basic program, so they are the ideal candidates for the destination pointer. We know that 2 and 3 will be free - they normally point to VAR-TOP - so we'll use those to point to the byte to be copied. In case you don't have a Master, we'll use the (zp),Y indexing mode.

The very last thing in the assembly text must be a label - it seems at this stage to refer to nothing, but in fact it labels what will eventually be the first byte of the Basic program which we're going to

store after our code. Our copying routine needs this address at the start, of course. Since I love witty labels, I wrote ".basProg" immediately before the final ']'. Right...

```
      .go LDY #0:STY 0 \ prepare Y and
destination index lo-byte.
      LDA &18:STA 1 \ copy hi-byte of PAGE to
dest. index hi-byte.
      LDA #basProg MOD &100:STA 2 \ set up
source index lo-byte.
      LDA #basProg DIV &100:STA 3 \ ditto
hi-byte.
      .loop LDA (2),Y:STA (0),Y \ shift a
byte from SRAM to normal area.
```

Whoopee, we're away! Oh... I forgot. Before we thoughtlessly write an INY:BNE loop, how are we going to know when we've copied the last byte? Well, Basic uses a 'negative hi-byte of line-number' end-marker, which in practice amounts to &0D followed by &FF. So...

```
      CMP #&0D:BNE inc \ if not &0D, go to
increment pointers.
      JSR upOne \ if it was &0D, go up one to
see if next is &FF.
      LDA (0),Y:STA (2),Y \ &FF or not, it
must be copied.
      CMP #&FF:BEQ copyDone. \ If it was &FF,
jump out of loop, else...
      .inc
      JSR upOne \ increment pointers...
      JMP loop \ and go round again.
      .upOne \ subroutine to update Y and
indexes.
      INY:BNE ret \ if Y isn't now zero,
indexes don't need updating.
      INC 1:INC 3 \ if we're here, Y was
zero. Update hi-bytes of indexes.
      .ret
      RTS
      .copyDone
```

That's the program copied across - how are we going to get it to run? We could (a) find the address to call in the Basic ROM; (b) insert the command RUN + Return into the keyboard buffer via the vector INSV at &022A or (c) insert same into the buffer using OSBYTE &8A. The OSBYTE is easiest, so now we need to include the string "RUN" in the program, to be copied into the buffer. We can't put it after *.copyDone*, of course; let's stick it before the *.helpText:*

```
      .runString EQUS "RUN":EQUB &0D:BRK \
BRK (= zero) is our end-marker.
```

Now back to the label *.copyDone:* next we write a loop to copy *.runString* into the buffer. With OSBYTE &8A all three registers are in use: A=&8A, of course, X=0 to specify which buffer (0 is the keyboard buffer number) and Y holds the character to be inserted. This means pushing the index register before each call:

```
      LDX #0 \ set up index register.
      .loop
      LDA #&8A \ OSBYTE number.
      LDY runString,X \ next character for
buffer goes into Y.
      BEQ hicCaestusArtemqueRepono \ if zero,
it was the end marker.
      PHX:LDX #0 \ save old X and point to
buffer number 0.
      JSR osbyte \ the dastardly deed is done
- we've poked it in.
      PLX:INX:JMP loop \ restore old X,
increment it & go round loop.
      .hicCaestusArtemqueRepono
      PLY:PLX:PLA:LDA #0:RTS \ claim call &
exit ROM
      .basProg
      ]:NEXT:ENDPROC
```

BBC B owners will have to put PLA:PLA:PLA. They should have altered

all the earlier pushes and pulls in the header text, too - see Beebug Vol.11 No.7 p.52.

Type all the above in, bung it in an assembly loop, save it, stick the Basic program on the end, and... it won't work. We haven't updated Basic's page zero pointers. Happily, the command OLD does this from scratch by whizzing through the program looking for the end-marker. We can add OLD to the RUN string. While we're at it, why not do *BASIC first, so the program can be called from, say, View. It's a bit primitive to select Basic *after* the program goes in, but it works from most other languages. So, let's alter *.runString* to:

```
EQUS"*B.":EQUB &0D:EQUS"OLD":EQUB
&0D:EQUS"RUN":EQUB &0D:BRK
```

You can now save this listing and run it. The complete listing, for lazy old toads, is shown below and has the added bonus of showing another way to create the *.runString* string.

P% now holds the address of the byte after the final RTS, so now do PRINT~P%, or use *.basProg* - same thing. It should be somewhere around &80C0. (All these numbers must be in Hex.) Now load your Basic program in the usual way and get the next numbers you'll need by PRINT~PAGE and PRINT~TOP. I get &1A48 for TOP with my Clock program; PAGE on a Master is almost always &E00. So, I now do:

```
*SRWRITE E00 1A48 80C0 7
```

assuming it's in SRAM slot 7 - the end routine in my header will have told you this. Now the Basic is in place; to save it you need the address of the end of the whole thing. Do PRINT~P%+TOP-PAGE. I get &8D08, and the start will always be &8000, so I now do:

```
*SRSAVE KLOCK 8000 8D08 7
```

and the whole thing is on disc under the name *KLOCK* (to avoid using the same filename as the original Basic.) Load and call it just like any other Sideways ROM. For a fun project, you could use my biggest ROM header and put several programs in, each called with a different star command. Modifying the code would be trivial, but getting all the addresses right could cause a few grey hairs.

This month's competition: (a) the four bytes CMP #&FF:BEQ copyDone can be reduced to two. How? (b) How does Mr T always get away with labelling every loop with the same label, *.loop?* Now come on, those are dead easy. There are still plenty of badges left.

Next month, a routine which tells you the number of shopping days to Christmas 1994. They're coming to take me away, ha ha....

```
  10 REM Basic program in SRROM format
  20 REM Version B 1.0
  30 REM Author  David Holton
  40 REM BEEBUG  December 1993
  50 REM Program subject to copyright
  60 :
 100 PROCassem
 110 FOR N%=7 TO 4 STEP-1
 120 IF N%?&2A1 NEXT:PRINT''"No free SR
AM slot."':END
 130 OSCLI "SRWRITE "+STR$~Z%+" "+STR$~
(O%+1)+" 8000 "+STR$ N%
 140 PRINT''"READY IN SLOT ";N%'"P% = &
";~P%
```

```
 150 N%?&2A1=&82:N%=4:NEXT:END
 160 :
1000 DEF PROCassem
1010 osasci=&FFE3
1020 osbyte=&FFF4
1030 FOR N%=4 TO 6 STEP 2
1040 Z%=?2+&100*?3-&200*(N%=4)
1050 P%=&8000:O%=Z%
1060 :
1070 [ OPT N%
1080 BRK:BRK:BRK
1090 JMP checkCalls
1100 EQUB &82
1110 EQUB copyright MOD &100
1120 EQUB &93
1130 EQUS "BASIC SRROM"
1140 .copyright
1150 BRK:EQUS"(C) BEEBUG 1993"
1160 :
1170 .checkCalls
1180 PHA:PHX:PHY
1190 CMP #4:BEQ isItOurs
1200 CMP #9:BNE noClaim
1210 :
1220 LDX #&FF
1230 .helpLoop
1240 INX:LDA helpText,X:PHP:JSR osasci
1250 PLP:BNE helpLoop
1260 :
1270 .noClaim
1280 PLY:PLX:PLA:RTS
1290 :
1300 .isItOurs
1310 LDX #0
1320 .checkLoop
1330 LDA starCommand,X:BEQ go
1340 LDA (&F2),Y:AND #&DF
1350 CMP starCommand,X:BNE noClaim
1360 INX:INY:JMP checkLoop
1370 :
1380 .helpText
1390 EQUW &0D0D: EQUS "BASIC SRROM"+CHR
$ &0D+"  "
1400 .starCommand
1410 EQUS"BAS":EQUB &0D:BRK
1420 .runString
1430 EQUS"*B."+CHR$ &0D+"OLD"+CHR$ &0D+
"RUN"+CHR$ &0D:BRK
1440 ::::::::::::::::::::::::
1450 .go
1460 LDY #0:STY 0
1470 LDA &18:STA 1
1480 LDA #basProg MOD &100:STA 2
1490 LDA #basProg DIV &100:STA 3
1500 :
1510 .loop
1520 LDA (2),Y:STA (0),Y
1530 CMP #&0D:BNE inc
1540 JSR upOne
1550 LDA (2),Y:STA (0),Y
1560 BMI allDone
1570 .inc
1580 JSR upOne:JMP loop
1590 :
1600 .upOne
1610 INY:BNE ret
1620 INC 1:INC 3
1630 .ret
1640 RTS
1650 :
1660 .allDone
1670 LDX #0
1680 :
1690 .loop
1700 LDA #&8A
1710 LDY runString,X
1720 BEQ hicCaestusArtemqueRepono
1730 PHX:LDX #0:JSR osbyte
1740 PLX:INX:JMP loop
1750 :
1760 .hicCaestusArtemqueRepono
1770 PLY:PLX:PLA:LDA #0:RTS
1780 .basProg
1790 ]:NEXT:ENDPROC
```

Public Domain

Alan Blundell looks at Master 512 software and a professionally-produced shareware product which upgrades the BBC micro to the level of a Z88.

An interesting item of shareware has come to my attention this month. It is *VP-Support* version 2.20 by David Lucas of Hailsham, Sussex. *VP-Support* is designed to emulate 'PipeDream', the integrated package from Colton Software on the Sinclair Z88 portable computer. It does this by enhancing the facilities of View Professional, which is a similar package (by the same authors) for the BBC Micro. The package is, therefore, only of use if you already own a copy of View Professional (View Professional is not the same as the View word processor, fitted as standard to the Master 128 and available for other 8-bit Acorn computers, in case there are any newer users who are unsure).

If you own both a BBC Micro with View Professional and a Z88 (with PipeDream), then VP-Support will no doubt be even more attractive, although if you aren't familiar with PipeDream, three help screens detail all of the available Z88 'star' commands (as distinct from Acorn's operating system in-built commands, also referred to as 'star' commands by way of shorthand).

VP-Support allows Z88 'star' commands to be entered using the Ctrl key and translates them into their View Professional equivalents. For example, the PipeDream command, <>PU is translated to \H1. *VP-Support* also provides windows for some commands, just as on the Z88. As a by-product, *VP-Support* also makes View Professional more compatible with the Archimedes version of PipeDream, since the latter uses the same commands. The program also provides full Z88 file transfer facilities, allows Z88 files to be archived on a BBC disc, and allows Z88 printer output to be channelled direct to a printer connected to the BBC micro.

The program is supplied as a 16K ROM image for a BBC micro, with an alternative version included for the Master 128. It should work with View Professional version 2.0 on a BBC Model 'B', (OS 1.2), with a Master 128, or with a 6502 Second Processor on either, and with DFS, ADFS and ANFS filing systems. According to David's comprehensive (on-disc) documentation, the program is compatible with Aries B-32 shadow RAM but may not work with other shadow RAM boards for the BBC 'B' if they use different access mechanisms. *VP-Support* uses sideways RAM as workspace to implement its windows interface, which has the advantage that it needs only minimal main memory workspace, and does not encroach at all on memory available for text storage. Sideways RAM used on a BBC 'B' must not be write-protected. On a Master, sideways RAM can't be write-protected anyway. Finally, *VP-Support* is designed only to work in modes 0 and 3, together with the equivalent shadow modes (128 and 131 on a Master). Other screen modes will result in a "Bad mode" error.

VP-Utilities is a variation on *VP-Support* which is also provided as a 16K ROM image, suitable for blowing into an EPROM if you have the facilities (or know someone who has). It contains *VP-Support* together with a number of additional utilities. *VP-Support* still needs to run in a sideways RAM bank; this option simply allows it to be loaded into sideways RAM from EPROM into any available vacant sideways RAM bank. The additional utilities are implemented

as Acorn style star commands this time, rather than the Z88 'star' key commands.

The new commands include, for example, *FMOVE, which transfers files between DFS, ADFS and ANFS, *RREMOVE and *RINSERT, equivalents to the Master's *UNPLUG/*INSERT commands (which are used to disable and re-enable particular sideways ROMs) but which also work on the Model 'B', and *RCLEAR to erase the contents of a sideways RAM bank. *VP-Support* itself may also be unhooked from View Professional by the *VPREMOVE command.

Altogether, this is a very professional and useful package, well worth the £5.00 shareware registration fee requested. Registration of shareware programs which you intend to make use of after a period of evaluation is a moral obligation, but in this case, it entitles you to any future improvements in the package as well as recompensing David for all the work which has obviously gone into it.

MASTER 512 SOFTWARE

I promised last issue that I would mention the latest Master 512 co-processor related software this issue (I also promised to begin a look at other sources of material for BBC users, but that will have to wait until next time). BBC PD disc GA3 is an 800K DOS format disc which contains hints on, and solutions to, a large number of commercial adventure games for DOS computers. Games covered include many Scott Adams adventures (some of which were made available in BBC Micro format many years ago), and the text based Infocom adventure game series - highly recommended if you are struggling with any of these games. Also on the disc are a couple of arcade type games, a text mode 'graphical' adventure game and an ESP experiment to while away your spare time on.

David Harper, whose name should by now be familiar as a source of expertly provided material for the 512, has recently come up with a 4 disc collection of PD and shareware which he has found to be compatible with the 512. There are no overlaps between the software on these discs and others which are currently available, although some of the software was previously in the Dabs Press collections which some of you may be aware of. David compiled the discs to complement the existing 21 discs in the BBC PD library, which already contained some but not all of the software which had appeared in those collections.

The collection of 4 discs contains over 5 megabytes of software, all of which is archived using PKZIP. David has actually used PKZIP version 2.04g, which needs a little setting up to work on the 512, but this is explained clearly in text files on the discs. I won't go into the relative merits of various versions of PKZIP as Robin Burton's *512 Forum* column has covered all of the relevant considerations. The contents of the discs are fully detailed in a large text file on the first of the four discs, but briefly they include lots of games, versions of Prolog and Basic programming languages, utilities, word processor, spelling and readability aids, world map program, cassette tape labeller utility, printing aids and more.

I won't go into more detail of the contents here, but I do think that David deserves recognition and thanks on behalf of those of us who still actively use the 512 co-pro for his efforts in compiling this collection, as well as for his previous contributions to its usefulness.

Next issue, with luck, I will begin the look at sources of software and support for the BBC which I had intended to find room for this issue.

Relocator

Miroslaw Bobrowski presents a utility to get things moving.

Relocator is a Basic program which allows you to change the start and execute addresses of any machine code file and to adjust appropriately all its internal addresses, i.e. addresses within the code. The only restriction is that the machine code processed must be relocatable within the parameters of the program and the target memory area should not include page &D. The listing below is for Master computers. Type it in and save it. When run it will give you step by step instructions.

Large machine code programs, like games and text editors, are unlikely be relocatable. This utility is helpful when dealing with short bits of code that do simple tasks. Perhaps you have several machine code utilities that you want resident in the machine at once; Relocate would allow you to load them in wherever you wanted them.

BUT WHAT IS RELOCATABLE?

Normally, the trick with relocatable code is that it contains no absolute references to itself. This means that no actual addresses are saved with the code. The only form of addressing you can use is relative addressing and that is only used by the branch instructions. When dealing with the accumulator you must use immediate mode but you can still call OS routines and use zero page as these have a fixed location.

This program extends that relocatability by reorganizing the program's internal addresses for you so internal JMPs, JSRs, LDAs etc. all now work, while OS calls are left alone. Obviously the program can't predict all the clever tricks you might try, so go slowly, but most eventualities are covered.

ON A BEEB

To use Relocator on a BBC B you will need to make the following changes to the code. Note that the file *Relocat* on the magazine disc is the Master version and so you will need to make these alterations to that.

```
 100 MODE 7
1580 EQUD &81810F81:EQUD &81178314
1600 EQUD &AAA999B9:EQUD &55655555
1610 EQUD &66AA5565:EQUD &EEEDEEEE
1620 EQUD &77F77775:EQUD &FFF7FFFF
```

```
  10 REM Program Relocator
  20 REM Version B 1.5
  30 REM Author  Miroslaw Bobrowski
  40 REM BEEBUG  December 1993
  50 REM Program subject to copyright
  60 :
 100 MODE 135:T%=TOP+&300
 110 DIM code% &D0,name% 20,block% 20
 120 PROCassemble
 130 FORI%=1TO2:PRINT TAB(6,I%)CHR$141C
HR$130CHR$157CHR$129"M/C file Relocator
 "CHR$156:NEXT
 140 PRINT''CHR$131"Enter name of file
to be relocated":REPEAT:PRINT TAB(5,6)SP
C120;TAB(5,6)CHR$130;:INPUT": "f$:F%=OPE
NIN(f$):IF F%=0 VDU7:PRINT'CHR$129"No su
ch file":XX=INKEY(200)
 150 UNTILF%<>0:CLOSE #0:$name%=f$:PROC
osfile5
 160 IF T%+L%>&8000 PRINT'CHR$129"File
too long to be processed":VDU7:END
 170 PRINT'CHR$131"Enter name for reloc
ated file"'TAB(5)CHR$130;:INPUT": "r$
 180 PRINT'CHR$131"Enter start address
for relocated"'CHR$131"file :"CHR$135;:I
NPUT"&"D$:D%=EVAL("&"+D$)
```

```
  190 IF D%<&D00 AND D%+L%>&D00 PRINT'CH
R$129"WARNING!"CHR$134"The resulting cod
e will"'CHR$134"overwrite page &D."
  200 IF D%+L%>&8000 PRINT'CHR$129"Desti
nation start address too high.":VDU7:END
  210 !&70=S%:!&72=D%:!&74=L%:!&80=T%:E%
=D%+(E%-S%)
  220 :
  230 OSCLI"LOAD "+f$+" "+STR$~T%
  240 CALL code%:IF r$="" PRINT'CHR$130"
Relocated code"CHR$129"not"CHR$130"saved
."':END
  250 PRINT'CHR$130"To save relocated fi
le press COPY ";:REPEAT UNTIL INKEY-106
  260 OSCLI"SAVE "+r$+" "+STR$~T%+" +"+S
TR$~L%+" "+STR$~E%+" "+STR$~D%
  270 PRINT STRING$(34,CHR$127)"Relocate
d code saved as ";CHR$34;r$;CHR$34'
  280 END
  290 :
 1000 DEF PROCassemble
 1010 srce=&70:dest=&72:len=&74
 1020 diff=&76:temp=&78:ptr=&7A
 1030 start=&7C:endaddr=&7E
 1040 addr=&80
 1050 FOR pass=0 TO 2 STEP 2
 1060 P%=code%
 1070 [OPT pass
 1080 LDA srce:STA temp
 1090 LDA srce+1:STA temp+1
 1100 LDA dest:STA start
 1110 CLC:ADC len:STA endaddr
 1120 LDA dest+1:STA start+1
 1130 ADC len+1:STA endaddr+1
 1140 SEC:LDA #0:SBC len:STA ptr
 1150 LDA #0:SBC len+1:STA ptr+1
 1160 SEC:LDA start:SBC temp:STA diff
 1170 LDA start+1:SBC temp+1:STA diff+1
 1180 .reloc
 1190 LDY #0:LDA (addr),Y
 1200 TAY:AND #&0F
 1210 TAX:LDA rdata,X
 1220 BMI update
 1230 TYA:ASL A
 1240 PHP
 1250 SEC:SBC rdata,X
 1260 PLP:ROL A:ROL A
 1270 TAY:AND #&1F
 1280 TAX:TYA
 1290 ROL A:ROL A
 1300 AND #3:TAY:LDA rdata2,X
 1310 .loop5
 1320 DEY:BMI update
 1330 LSR A:LSR A:BNE loop5
 1340 .update
 1350 AND #3:CMP #3:BNE update2
 1360 PHA
 1370 LDY #1:LDA (addr),Y
 1380 CLC:ADC diff:STA temp
 1390 TAX:INY:LDA (addr),Y
 1400 ADC diff+1:STA temp+1
 1410 CPX start:SBC start+1:BCC update1
 1420 LDA endaddr:CMP temp
 1430 LDA endaddr+1:SBC temp+1:BCC updat
e1
 1440 LDA temp+1:STA (addr),Y
 1450 DEY:TXA:STA (addr),Y
 1460 .update1
 1470 PLA
 1480 .update2
 1490 CLC:ADC addr:STA addr
 1500 BCC adjust_pointer
 1510 INC addr+1
 1520 .adjust_pointer
 1530 INC ptr:BNE reloc
 1540 INC ptr+1:BNE reloc
 1550 RTS
 1560 .rdata
 1570 EQUD &81038200:EQUD &81828206
 1580 EQUD &81810F81:EQUD &81838314
 1590 .rdata2
 1600 EQUD &AAAA99B9:EQUD &99A99999
 1610 EQUD &66AAA5AA:EQUD &EEEEEEEE
 1620 EQUD &77FFF7FF
 1630 :
 1640 ]:NEXT
 1650 ENDPROC
 1660 :
 1670 DEF PROCosfile5
 1680 !block%=name%
 1690 X%=block% MOD 256
 1700 Y%=block% DIV 256
 1710 A%=5:CALL &FFDD
 1720 S%=block%!2
 1730 E%=block%!6
 1740 L%=block%!10
 1750 ENDPROC
```

Pyramid Patience

Leslie Fowl tries to keep us up all night.

This is a particularly difficult version of the old card game which should keep you busy over the long winter evenings ahead. Type in the programs *Pyramid* shown below and save it. Subscribers to the magazine disc will find the program in two parts, *Pyramid* and *Pyr* to include on-screen instructions.

Running Pyramid presents a playing screen as shown in figure 1.

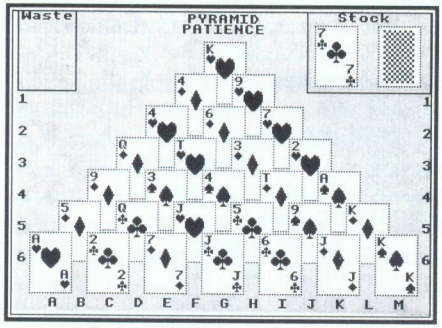


Figure 1. The starting point

THE GAME PLAN

The object of the game is to remove all the cards from the pyramid, one pair at a time, in pairs that total thirteen, including all the Stock and Waste cards, but with the exception of the apex King. Cards have their expected values with Jack, Queen and King being 11, 12 and 13 respectively. Before you play you can select the type of shuffling the program will do, this will give you different levels of difficulty.

PLAY THE GAME

At the bottom of the screen is the input and error message line. The first card co-ordinates are entered at the prompt in the form of 1A or 2C etc. followed by Return. Next you are prompted for card number two (assuming that the first card was not a King, as King's on their own are valued at thirteen). Again enter the card co-ordinates or in either case you may enter 'S' or 'W' followed by Return if you want to include the card from either the Stock or Waste Piles. Initially the only cards available for play are those in the bottom row and the first stock card. At any time during play press 'T' to reveal the number of cards left in both Waste & Stock piles. Pressing 'Q' ends the game. To deal the Stock card press '@'; this transfers the stock card to the waste pile (still face up and still available for play) and reveals the next stock card. The other cards in the the Pyramid become available only when the two adjacent cards below are removed. All the above key presses must be followed by Return.

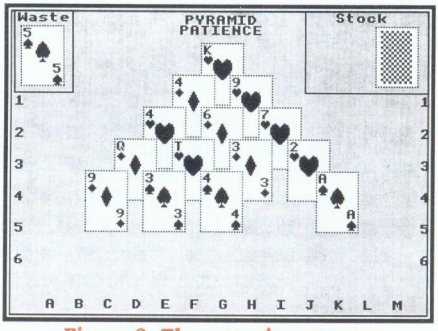


Figure 2. The game in progress

The only hint or tip I can give is not to allow the Waste Pile to become too big if at all possible.

When the game is eventually completed the player is rewarded with a pleasant little tune, I hope you hear it often - good luck.

```
  10 REM Program Pyramid
  20 REM Version B 1.0
  30 REM Author  Leslie Fowl
  40 REM BEEBUG  December 1993
  50 REM Program subject to copyright
  60 :
 100 ON ERROR MODE7:VDU7:REPORT:PRINT;"
at line ";ERL:END
 110 DIM cx%(30),cy%(30),p%(30),cv$(30)
 120 DIMip$(2),cval%(2),pi%(22),du%(22)
 130 DIM rl$(6):game%=0
 140 PROCpeter
 150 CLS:FOR Y%=12 TO 13:PRINTTAB(11,Y%
);CHR$141+CHR$134+"Shuffling..":NEXT
 160 PROCinit:PROCshuffle
 170 IF SH% PROCshuffle
 180 temp$=apex$+f$
 190 VDU22,129:PROCtable
 200 PROCtableu:temp$="":PROCdeal
 210 PROCplay:CLS
 220 IF kflag GOTO 250
 230 PROCremovecard(c%,d%)
 240 IF st AND wa temp$=""
 250 PROCremovecard(A%,B%)
 260 IF st tflag=TRUE:PROCdeal
 270 IF wa PROCupdatewaste:VDU4
 280 PROCcheckwin
 290 IF win=FALSE GOTO210
 300 END
 310 :
1000 DEF PROCtableu:r=0
1010 REPEAT X%=X%-88:Y%=Y%-96:x=X%:y=Y%
1020 FORx=x TO X%+step%*r STEP step%:c%
=c%+1
1030 cx%(c%)=x:cy%(c%)=y:p%(c%)=c%
1040 PROCbrdr(x,y):GCOL0,3
1050 VDU5:MOVEx,y:PLOT97,120,step%
1060 PROCdisplay(x,y,temp$,c%)
1070 temp$=RIGHT$(temp$,LEN(temp$)-2)
1080 NEXT:r=r+1:UNTIL r=7:card$=""
1090 ENDPROC
1100 :
1110 DEF PROCbrdr(x,y):GCOL0,0
1120 MOVEx-2,y-2:PLOT29,x-2,y+180
1130 PLOT29,x+124,y+180:PLOT29,x+124,y-
2
1140 PLOT29,x-2,y-2:ENDPROC
1150 :
1160 DEF PROCtable:VDU5
1170 VDU28,0,31,39,29,24,0;100;1279;102
3;
1180 VDU19,2,2,0,0,0:GCOL0,130:CLG
1190 CLS:GCOL0,0:N=64
1200 MOVE524,1008:PRINT"PYRAMID"
1210 MOVE508,968:PRINT"PATIENCE"
1220 FOR x=104 TO 1228 STEP 88:N=N+1
1230 MOVE x,140:PRINT;CHR$(N):NEXT
1240 N=0:FOR y=756 TO 276 STEP-96
1250 N=N+1:MOVE8,y:PRINT;N:MOVE1244,y:P
RINT;N:NEXT
1260 MOVE8,1016:DRAW184,1016:DRAW184,76
8:DRAW8,768:DRAW8,1016:MOVE16,1008:VDU5:
PRINT"Waste"
1270 MOVE894,1016:DRAW1270,1016:DRAW127
0,768:DRAW894,768:DRAW894,1016:MOVE986,1
008:PRINT"Stock"
1280 MOVE216,1008:PRINT"Game ";game%
1290 ENDPROC
1300 :
1310 DEF PROCsuits(card$)
1320 IF RIGHT$(card$,1)="H"THEN suit$=H
$:suit2$=H2$:GCOL0,1
1330 IF RIGHT$(card$,1)="D"THEN suit$=D
$:suit2$=D2$:GCOL0,1
1340 IF RIGHT$(card$,1)="C"THEN suit$=C
$:suit2$=C2$:GCOL0,0
1350 IF RIGHT$(card$,1)="S"THEN suit$=S
$:suit2$=S2$:GCOL0,0
1360 ENDPROC
1370 :
1380 DEF PROCdisplay(x,y,c$,c%)
1390 card$="":card$=LEFT$(c$,2)
1400 cv$(c%)=card$
1410 pip$=LEFT$(cv$(c%),1)
1420 DEF PROCdisplay1(card$)
1430 IF card$="" ENDPROC
1440 PROCsuits(card$)
1450 MOVEx+2,y+172:PRINTpip$
1460 MOVEx+2,y+140:PRINTsuit$
1470 MOVEx+32,y+132:PRINTsuit2$
1480 MOVEx+88,y+64:PRINTpip$
1490 MOVEx+88,y+32:PRINTsuit$
```

```
 1500 SOUND0,-10,4,1:ENDPROC
 1510 :
 1520 DEF PROCstock(x,y):GCOL0,3
 1530 IF S%=0 tflag=FALSE:ENDPROC
 1540 MOVEx,y:MOVEx,y-172
 1550 PLOT85,x+120,y
 1560 MOVEx+120,y-172:PLOT85,x,y-172
 1570 VDU5:MOVE1124,958:GCOL0,1:PRINTSB$
 1580 PROCbrdr(x,y-176):ENDPROC
 1590 :
 1600 DEF PROCshuffle
 1610 f$="":D%=51:FORI%=1TO51
 1620 A%=INT(RND(RND(1))*D%+1)
 1630 f$=f$+MID$(temp$,2*A%-1,2)
 1640 L$=LEFT$(temp$,(A%-1)*2)
 1650 R$=RIGHT$(temp$,(LEN(temp$)/2-A%)*
2)
 1660 temp$=L$+R$:D%=D%-1
 1670 NEXT I%:temp$=f$
 1680 stock$=RIGHT$(temp$,48)
 1690 IF PP%=FALSE ENDPROC
 1700 FOR I%=1 TO LEN(stock$)-1 STEP 2
 1710 IF MID$(stock$,I%,1)=MID$(stock$,I
%+2,1) THEN stock$=stock$+MID$(stock$,I%
+2,2):stock$=LEFT$(stock$,I%+1)+MID$(sto
ck$,I%+4,LEN(stock$)-(I%+2))
 1720 NEXT ELSE NEXT
 1730 ENDPROC
 1740 :
 1750 DEF PROCinit
 1760 S%=25:W%=0:tflag=FALSE:game%=game%
+1
 1770 cx%(0)=1112:cy%(0)=790
 1780 cx%(29)=32:cy%(29)=790
 1790 cx%(30)=920:cy%(30)=790
 1800 pack$="A23456789TJQK":waste$="":W$
=""
 1810 key$="ABCDEFGHIJKLMWS@TQ":stock$="
"
 1820 suit$="HCDS":X%=668:Y%=832:step%=1
76
 1830 C$="":apex$="":temp$="":waste2$=""
 1840 rl$(1)="FH":rl$(2)="EGI"
 1850 rl$(3)="DFHJ":rl$(4)="CEGIK"
 1860 rl$(5)="BDFHJL":rl$(6)="ACEGIKM"
 1870 count%=28:r%=0:c%=0:SH%=TRUE:PP%=T
RUE
 1880 FOR I%=1 TO 13:FOR J%=1 TO 4
 1890 temp$=temp$+MID$(pack$,I%,1)+MID$(
suit$,J%,1)
 1900 NEXT J%:NEXT I%
 1910 apex$=RIGHT$(pack$,1)+(MID$(suit$,
INT(RND(4)),1))
 1920 FOR k%=1 TO LEN(temp$) STEP2
 1930 IF MID$(temp$,k%,2)=apex$ THEN NEX
T ELSE C$=C$+MID$(temp$,k%,2):NEXT
 1940 C$=apex$+C$:temp$="":temp$=RIGHT$(
C$,LEN(C$)-2)
 1950 D$=CHR$(231):H$=CHR$(232)
 1960 C$=CHR$(233):S$=CHR$(234)
 1970 nl$=CHR$(8)+CHR$(8)+CHR$(10)
 1980 D2$=CHR$(235)+CHR$(236)+nl$+CHR$(2
37)+CHR$(238)
 1990 H2$=CHR$(239)+CHR$(240)+nl$+CHR$(2
41)+CHR$(242)
 2000 C2$=CHR$(243)+CHR$(244)+nl$+CHR$(2
45)+CHR$(246)
 2010 S2$=CHR$(247)+CHR$(248)+nl$+CHR$(2
49)+CHR$(250)
 2020 SB$="":sb$=CHR$230+CHR$230+CHR$230
+CHR$8+CHR$8+CHR$8+CHR$10
 2030 FOR I%=1TO5:SB$=SB$+sb$:NEXT
 2040 RESTORE
 2050 FOR I%=1TO22:READN,T
 2060 pi%(I%)=N:du%(I%)=T:NEXT
 2070 DATA117,5,129,5,129,10,117,5,109,5
 2080 DATA101,10,109,5,117,5,129,5,117,5
 2090 DATA109,20,117,5,129,5,129,10,117,
5
 2100 DATA109,5,101,10,109,5,117,5,109,5
,101,5,101,20
 2110 ENDPROC
 2120 :
 2130 DEF PROCplay
 2140 CLS:C%=0:I%=0:*FX15,1
 2150 wa=FALSE:st=FALSE:kflag=FALSE
 2160 FOR I%=1TO2
 2170 PRINT"Enter card #";I%;:IF (S%<2 A
ND I%<2) PRINT;" or Q to quit";
 2180 INPUT" "ip$(I%)
 2190 C%=INSTR(key$,RIGHT$(ip$(I%),1))
 2200 IF C%=0 PROCerr(2):I%=2:NEXT:GOTO2
```

```
140
 2210 IF C%=16 PROCdeal:I%=2:NEXT:GOTO21
40
 2220 IF C%=17 PROCerr(7):I%=2:NEXT:GOTO
2140
 2230 IF C%=18 I%=2:NEXT:PROCquit:ENDPRO
C
 2240 IF C%=14 B%=29:A%=0:wa=TRUE:GOTO23
40
 2250 IF C%=15 B%=30:A%=0:st=TRUE:GOTO23
40
 2260 A%=VAL(LEFT$(ip$(I%),1))
 2270 row%=FNconvert(A%)
 2280 IF row%=0 PROCerr(1):GOTO2140
 2290 IF NOT FNrl(ip$(I%)) THEN PROCerr(
8):I%=2:NEXT:GOTO2140
 2300 B%=ASC(RIGHT$(ip$(I%),1))/2-row%
 2310 IF (p%(B%)=0 AND cv$(B%)="")PROCer
r(3):CLS:GOTO 2140
 2320 IF A%<6:IF (p%(B%+(2+A%))<>0 OR p%
(B%+(A%+1))<>0) PROCerr(4):GOTO2140
 2330 IF C%>13 AND C%<16 cv$(B%)=temp$
 2340 cval%(I%)=FNval(LEFT$(cv$(B%),1))
 2350 IF cval%(I%)=13 THEN kflag=TRUE:I%
=2:NEXT:ENDPROC
 2360 IF I%=1 c%=A%:d%=B%
 2370 NEXT I%
 2380 IF cval%(1)+cval%(2)<>13 PROCerr(5
):GOTO2140
 2390 ENDPROC
 2400 :
 2410 DEF FNconvert(A%)
 2420 IF (A%<1 OR A%>6):=FALSE
 2430 IF A%=1:=33 ELSE IF A%=2:=30
 2440 IF A%=3:=27 ELSE IF A%=4:=22
 2450 IF A%=5:=17 ELSE IF A%=6:=10
 2460 :
 2470 DEF FNval(V$)
 2480 IF V$="A":=1 ELSE IF V$="T":=10
 2490 IF V$="J":=11 ELSE IF V$="Q":=12
 2500 IF V$="K":=13 ELSE =VAL(V$)
 2510 :
 2520 DEF FNrl(ip$(I%))
 2530 r=VAL(LEFT$(ip$(I%),1))
 2540 c$=RIGHT$(ip$(I%),1)
 2550 IF INSTR(rl$(r),c$) THEN =TRUE ELS
E =FALSE
 2560 :
 2570 DEF PROCremovecard(a%,b%)
 2580 GCOL0,2
 2590 MOVEcx%(b%)-2,cy%(b%)-2
 2600 PLOT97,126,182
 2610 cv$(b%)="":p%(b%)=0
 2620 IF C%>13 AND S%>0 tflag=TRUE
 2630 IF a%=0 THEN ENDPROC
 2640 SOUND0,-10,4,1
 2650 PROCcheck(a%,b%)
 2660 PROCrepair(a%,b%)
 2670 ENDPROC
 2680 :
 2690 DEF PROCrepair(A%,B%)
 2700 I%=(B%-(A%+1))
 2710 IF NOT p%(I%) GOTO2790
 2720 GCOL0,3:MOVEcx%(I%)+86,cy%(I%)
 2730 PLOT97,34,84
 2740 MOVEcx%(I%)+86,cy%(I%)-2:GCOL0,0
 2750 PLOT29,cx%(I%)+124,cy%(I%)-2
 2760 PLOT29,cx%(I%)+124,cy%(I%)+88
 2770 IF p%(I%)<>0 THEN p%(I%)=I%
 2780 PROCdisplay2(I%)
 2790 I%=(B%-A%):GCOL0,3
 2800 IF NOT p%(I%) ENDPROC
 2810 MOVEcx%(I%),cy%(I%):PLOT97,38,86
 2820 GCOL0,0:MOVEcx%(I%)-2,cy%(I%)+88
 2830 PLOT29,cx%(I%)-2,cy%(I%)-2
 2840 PLOT29,cx%(I%)+40,cy%(I%)-2
 2850 IF p%(I%)<>0 THEN p%(I%)=I%
 2860 ENDPROC
 2870 :
 2880 DEF PROCcheck(A%,B%)
 2890 IF (B%=2 OR B%=4 OR B%=7 OR B%=11
OR B%=16 OR B%=22) THEN p%(B%-A%)=TRUE:E
NDPROC
 2900 IF (B%=3 OR B%=6 OR B%=10 OR B%=15
 OR B%=21 OR B%=28) THEN p%(B%-(A%+1))=T
RUE:ENDPROC
 2910 IFp%(B%-A%)<>0 THEN p%(B%-A%)=TRUE
 2920 IF p%(B%-(A%+1))<>0 THEN p%(B%-(A%
+1))=TRUE
 2930 ENDPROC
 2940 :
 2950 DEF PROCdisplay2(I%)
```

```
 2960 pip$=LEFT$(cv$(I%),1)
 2970 card$=cv$(I%):PROCsuits(card$)
 2980 MOVEcx%(I%)+88,cy%(I%)+64
 2990 VDU5:PRINTpip$
 3000 MOVEcx%(I%)+88,cy%(I%)+32
 3010 PRINTsuit$:VDU4:ENDPROC
 3020 :
 3030 DEF PROCerr(e%):COLOUR1:*FX15,1
 3040 SOUND1,-15,20,5:CLS:PRINT
 3050 IF e%=1 PRINT"row number incorrect
 ";
 3060 IF e%=2 PRINT;"Invalid entry.";
 3070 IF e%=3 PRINT;ip$(I%);" Has alread
y been removed.";
 3080 IF e%=4 PRINT;ip$(I%);" Not yet av
ailable.";
 3090 IF e%=5 PRINT;ip$(1);"+";ip$(2);"
Do NOT total 13.";
 3100 IF e%=6 PRINT;"Stock exhausted.";
 3110 IF e%=7 PRINT;"Cards remaining :"'
'"Waste=";W%;"  Stock=";S%;" ";
 3120 IF e%=8 PRINT"Col to Row input mis
-match";
 3130 COLOUR2:PRINT" HIT SPACE"
 3140 REPEAT UNTIL GET=32:CLS
 3150 COLOUR3:CLS:e%=0:ENDPROC
 3160 :
 3170 DEF PROCdeal
 3180 IF (S%=1 AND C%=16) THEN S%=1:PROC
err(6):ENDPROC
 3190 S%=S%-1
 3200 IF S%<=0 S%=0:PROCerr(6):ENDPROC
 3210 IF S%=1 PROCremovecard(0,0):GOTO32
30
 3220 PROCstock(1112,966)
 3230 IF (S%>=0 AND S%<=23 AND W%>=0 AND
 C%=16) tflag=FALSE
 3240 IF tflag temp$=""
 3250 IF W%<=0 W%=0
 3260 W$=W$+temp$
 3270 IF S%>0 AND S%<24 AND NOT tflag W%
=W%+1
 3280 temp$=LEFT$(stock$,2):cv$(30)=temp
$
 3290 x=920:y=790:GCOL0,3
 3300 MOVEx,y:MOVEx+120,y
 3310 PLOT85,x,y+176:MOVEx+120,y+176
 3320 PLOT85,x+120,y:PROCbrdr(x,y)
 3330 pip$=LEFT$(temp$,1):VDU5
 3340 PROCdisplay1(temp$):SOUND0,-10,4,1
 3350 stock$=RIGHT$(stock$,LEN(stock$)-2
)
 3360 PROCdispwaste(W$)
 3370 tflag=FALSE:VDU4:ENDPROC
 3380 :
 3390 DEF PROCdispwaste(W$)
 3400 waste$=RIGHT$(W$,2)
 3410 IF waste$="" OR waste$=waste2$ END
PROC
 3420 cv$(29)=waste$
 3430 x=32:y=790:GCOL0,3
 3440 MOVEx,y:MOVEx+120,y:PLOT85,x,y+176
 3450 MOVEx+120,y+176:PLOT85,x+120,y
 3460 PROCbrdr(x,y)
 3470 pip$=LEFT$(waste$,1):VDU5
 3480 PROCdisplay1(waste$):tflag=FALSE
 3490 waste2$=waste$
 3500 ENDPROC
 3510 :
 3520 DEF PROCupdatewaste
 3530 W%=W%-1:IF W%<0 W%=0
 3540 W$=LEFT$(W$,LEN(W$)-2)
 3550 PROCdispwaste(W$):ENDPROC
 3560 :
 3570 DEF PROCcheckwin
 3580 LOCAL I%:win=TRUE:count%=28
 3590 FOR I%=28 TO 2 STEP-1
 3600 IF p%(I%)>0 win=FALSE:I%=2:NEXT:EN
DPROC ELSE NEXT
 3610 IF W%>0 OR S%>0 win=FALSE:I%=2:END
PROC
 3620 GCOL0,0:VDU5
 3630 MOVE186,200:PLOT97,914,400
 3640 GCOL0,3:MOVE400,564:PRINT"CONGRATU
LATIONS"
 3650 MOVE 352,460:PRINT"You have achiev
ed"
 3660 MOVE 316,424:PRINT"The almost impo
sible"
 3670 FOR I%=1 TO 22
 3680 SOUND1,-15,pi%(I%),du%(I%)
 3690 SOUND1,0,0,1:NEXT
```

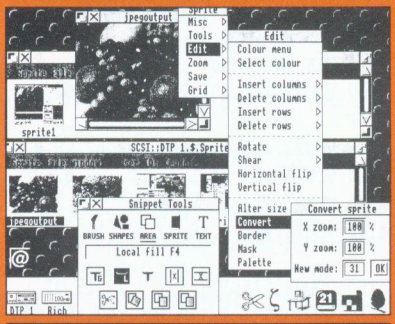
```
 3700 MOVE 196,324:PRINT"Would you like
to play again"
 3710 MOVE 512,264:PRINT"(Y/N)";
 3720 PROCyn:ENDPROC
 3730 :
 3740 DEF PROCquit:LOCAL I%
 3750 FOR I%=28 TO1 STEP-1
 3760 IF p%(I%)=0 count%=count%-1
 3770 NEXT:VDU22,7,7
 3780 PRINT'''"Bad Luck...."'"You have b
een unable to complete"'"Game ";game%:PR
INT'''"There are ";W%;" cards left in the
 waste."
 3790 PRINTTAB(10);S%;" cards left in st
ock."
 3800 PRINTTAB(6)"and ";count%;" cards l
eft in the pyramid."
 3810 PRINT'''"Better luck next time."''"
Play again ?  (Y/N)";
 3820 PROCyn:ENDPROC
 3830 :
 3840 DEF PROCyn
 3850 REPEAT G=INSTR("YyNn",GET$):UNTILG
>0 AND G<5
 3860 IF G<3 VDU22,7:GOTO140 ELSEVDU22,7
 3870 FOR Y%=3 TO 4:X%=0
 3880 PRINTTAB(X%,Y%);CHR$131CHR$141"Tha
nk you for playing":NEXT
 3890 FOR Y%=6 TO 7
 3900 PRINTTAB(X%,Y%);CHR$133CHR$141"Pyr
amid Patience.":NEXT:END
 3910 :
 3920 DEF PROCpeter
 3930 PROCvdus:CLS:X%=7:FOR Y%=1TO2
 3940 PRINTTAB(X%,Y%)CHR$134CHR$141"Pyra
mid  Patience":NEXT
 3950 FOR Y%=4TO5:X%=8
 3960 PRINTTAB(X%,Y%)CHR$131CHR$141"Shuf
fle Options":NEXT
 3970 PRINT''CHR$131"1)  Straight Shuffl
e only."
 3980 PRINT'CHR$131"2)  Shuffle + pairs
seperation."
 3990 PRINT'CHR$131"3)  Double Shuffle o
nly."
 4000 PRINT'CHR$131"4)  Double Shuffle +
 pairs seperation."
 4010 PRINT'"Pairs seperation applies to
 stock only."
 4020 PRINT'''CHR$130"Enter option numbe
r :";
 4030 REPEAT:G%=GET-48:UNTIL G%>0 AND G%
<5
 4040 IF G%=1 SH%=FALSE:PP%=FALSE
 4050 IF G%=2 SH%=FALSE:PP%=TRUE
 4060 IF G%=3 SH%=TRUE:PP%=FALSE
 4070 IF G%=4 SH%=TRUE:PP%=TRUE
 4080 CLS:ENDPROC
 4090 DEF PROCvdus
 4100 VDU23,230,204,204,51,51,204,204,51
,51
 4110 VDU23,231,8,28,62,127,62,28,8,0
 4120 VDU23,232,54,127,127,127,62,28,8,0
 4130 VDU23,233,8,28,28,107,127,107,8,28
 4140 VDU23,234,8,28,62,127,127,127,28,6
2
 4150 VDU23,235,1,3,3,7,7,15,15,31
 4160 VDU23,236,0,128,128,192,192,224,22
4,240
 4170 VDU23,237,31,15,15,7,7,3,3,1
 4180 VDU23,238,240,224,224,192,192,128,
128,0
 4190 VDU23,239,24,124,126,255,255,255,2
55,255
 4200 VDU23,240,48,124,252,254,254,254,2
54,254
 4210 VDU23,241,255,127,127,63,31,15,7,3
 4220 VDU23,242,254,252,252,248,240,224,
192,128
 4230 VDU23,243,3,7,15,15,15,7,59,125
 4240 VDU23,244,128,192,224,224,224,192,
184,124
 4250 VDU23,245,255,255,255,125,57,1,3,7
 4260 VDU23,246,254,254,254,124,56,0,128
,192
 4270 VDU23,247,1,3,7,15,15,31,31,63
 4280 VDU23,248,0,128,192,224,224,240,24
0,248
 4290 VDU23,249,63,63,31,31,13,1,3,7
 4300 VDU23,250,248,248,240,240,96,0,128
,192
 4310 ENDPROC
```

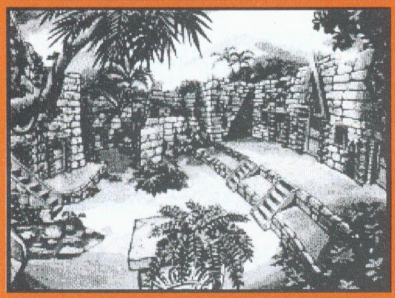
RISC user

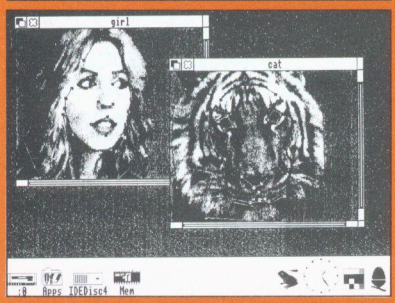
The number one subscription magazine for the Archimedes

RISC User, probably the most popular subscription magazine for the Archimedes, offers all the information you need as an Archimedes user. In every issue of RISC User you will find a wealth of articles and programs with professionally written reviews, lively news, help and advice for beginners and experienced users, and items of home entertainment.

The B5 size of RISC User allows a sophisticated design, big colour illustrations and pages full of information, and yet is still a convenient size to assemble into an easy-to-use reference library. Altogether, in its six years of existence, RISC User has established a reputation for a professional magazine with accurate, objective and informed articles of real practical use to all users of Acorn's range of RISC computers.







Contents of the latest Vol.7 Issue 2 of RISC User:

GROUP SURVEY: IMAGE PROCESSORS

A comprehensive round up of image processing software from public domain to Revelation imagePro.

INTRODUCING PHOTO-CD

An introductory article explaining all about the new Kodak Photo-CD standard now being implemented on Acorn computers.

CRYSTAL MAZE REVIEW

This is the game, based on the popular TV series, that's been attracting all the attention recently in the Acorn world.

CLARES' RHAPSODY 3

A review of the latest and most comprehensive music package yet providing a high quality music editing and printing application.

THE RISC USER MORPHER

A description of morphing, the latest craze in visual manipulation on the Archimedes, with a complete morphing application and demo on the magazine disc.

WIMP TOPICS

A major series aimed at readers interested in Wimp programs and Wimp programming. Each article looks at aspects of a particular topic.

WRITE-BACK

The readers' section of RISC User for comment, help, information - a magazine version of a bulletin board.

INTO THE ARC

A regular series for beginners.

TECHNICAL QUERIES

A column which answers your technical queries.

THE DOS SURVIVAL GUIDE

A series of articles on how to use the PC Emulator.

SUBSCRIPTION DETAILS

As a member of BEEBUG you may subscribe to RISC User for the reduced rate of £18.40 (a saving of £1.50 on the normal subscription rate).

Overseas subscription rates are as follows:
£27.50 Europe and Eire, £33.50 Middle East, £36.50 Americas & Africa, £39.50 Elsewhere

SINGLE KEY BAD PROGRAM RECOVERER

M.C.Behrend

The following function key definition may be able to help you recover your latest masterpiece after it has crashed with a "Bad Program" error message.

```
*KEY0 M%=PA.:?M%=13:M%?1=0:REP.REP.N%=M%+
3:REP.N%=N%+1:U.?N%<32ORN%-M%>250:?N%=13:
M%?3=N%-M%:P.M%?1*256+M%?2,~M%:M%=N%:
U.M%?1>127ORINKEY0<>-1:M%?1=M%?1OR128:
P."Further ?": G%=GET:IFG%=89ORG%=121
M%?1=M%?1A.127:U.FA.|M
```

FUNCTION KEY LISTING OF ENVELOPES

T.J.Young

Here is a useful function key definition that will display the parameters of any desired sound envelope. These are listed in the same order as in the ENVELOPE instruction, but you must supply the envelope number in response to the prompt:

```
*KEY0 MO.7:INP."Number of envelope ";N%:
@%=0: P."Envelope ";N%;:F.t%=0 TO
12:P.",";:E%=?(&8C0 +(N%-1)*16+t%):IF(t%>0
AND t%<4) OR (t%>6 AND t%<11) THEN IF E%>127
E%=E%-1:T%=E%EOR&FF:
P.T%;:N.:P.:@%=10:EL.T%=E%:P.T%;:N.:@%=10|M
```

QUICK SCREEN FILL

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

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

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

CONTEMPORARY IMPROVISATIONS

R.Tobin

Type G.0:G.0:G.0:G.0:.....etc., until you have filled the keyboard buffer - about 6 lines in mode 7. Then hit Return. You will of course get the *No such line* error, but it also has other effects.

PERSONALISED HEADER ON BREAK

J.Martins

There are of course more useful ways to use the Break vector, but the short piece of code below will personalise your title banner on Break.

```
 10 osasci=&FFE3:osbyte=&FFF4:PROCassemble
 20 CALL init:END
 30 DEF PROCassemble
 40 FOR pass%=0 TO 3 STEP 3:P%=&C00
 50 [OPT pass%:.start BCC exit:LDX#11
 60 LDY#0:.print LDAmess,Y:JSRosasci:INY
 70 DEX:BNEprint:.exit RTS
 80 .init LDY#0:LDA#&F7:LDX#&4C:JSRosbyte
 90 LDA#&F8:LDX#start MOD 256:JSRosbyte
100 LDA#&F9:LDX#start DIV 256:JSRosbyte
110 RTS:.mess:]
120 $P%="Your name":P%=P%+9
130 ?P%=13:P%?1=13:P%=P%+2
140 NEXT:ENDPROC
```

Simply replace the text "Your name", at line 120, with a suitable string no longer than nine characters. If you use this routine on an Econet system, then you should modify the address &C00 at line 40 to &A00, as Econet uses Page &C00 as its workspace.

TO LINEFEED OR NOT TO LINEFEED

Tim Dawe

If you find that, when using the View Printer Driver, BEEBUG Vol.12 No.4, your text is printed with double line spacing, here is a simple solution that does not involve setting DIP switches on your printer. The problem is caused by the printer driver sending both a carriage return and a linefeed at the end of each line, and the printer adding an extra linefeed itself. Make the following changes to the driver program:

```
105 oswrch=&FFEE
1140 JSR oswrch
1150 PLA:JMP oswrch
```

The printer driver will now only send a carriage return after each line, with the printer adding the linefeed. B

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 40p per word (inclusive of VAT) and these will be featured separately. Please send all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS.

Spectrum emulator, data transfer on 232/432i/o port, screen module, load and save in Spectrum Basic but executed on Acorn DFS and vice versa, a printer option is built in, also sound facility. Write to; Carsten Witt, Rostockerstr. 5, D 45739 Oer-Erkenschwick, Germany.

BBC software (Master 128) PMS Publisher ROM with manual and Font discs £20, Cambridge Micro Software Image with manual £15, ESM/Tedimen Advanced Folio (ADFS) with manual £15, GSN/ITV Key database (ADFS) with manual £8, Mid-Sussex Software Music Processor £15, Forth Dimension Holed Out £4, Acorn User discs April '91 to April '92 5.25" £3, May '92 to March '93 3.5" £3, Dabs Press Shareware Collection Vol.1 £10, Vol.2 £10, Profit Systems Tascmaster £10. Tel. Somerset 0458 43906.

Quantity of new 2764, 2732, 6264LP, please ring for details with sensible offers. Tel. Lancashire 0254 701573.

BBC model B with Acorn DFS, speech ROMs and Watford sideways ROM board, twin Cumana s/s disc drives, Prism modem and philips amber monitor, disc/tape educational/games software, disc box, blank discs and various books, all for £125 + p&p. Tel. Chester 0244 344695.

Computer Concepts RAM/ROM board fitted with 32k battery backed RAM chip, Interword, Intersheet and Spellmaster chips, complete with manuals and keystrips £80 o.n.o. Acorn original Maths co-processor podule £150 o.n.o. Tel. Cheshire 0925 811420.

BEEBUG magazines; March '84 - Aug '87 inclusive 35 back copies. Micro User Nov '84 - Dec '89 inclusive 61 back copies. Acorn User mags various April '84 - Feb '88 29 back copies. Telelink mags first issues Vol. 1-7 Nov '84 - Dec '85 and Vol. 2 1-2 Nov '86 - June '87. Tel. Perthshire 0764 670674.

BBC Master, Welcome and Reference Guides 1&2, 40/80T welcome discs, Viewspell ROM and manual, View manual, Cumana 40/80T disc drive and manual, Microvitec Cub 1451/DS2 colour monitor, Marconi tracker ball RB2 £300. Tel. Stirlingshire 0324 715586.

BBC B issue 7 with DFS, Cumana disc drive, Wordwise ROM and Epson MX100/3 printer, full working order £150. o.n.o. for the lot or might separate. Tel. Kent 0634 685492.

Fast Access volume 1 £15, Fast Access Volume 2 £20, Fast Access Volume 3 £12, Disc User disc set £14, Superior Software Spycat £2, Opus disc interface £10, Fontstyle printing package plus fonts £10, Viglen ROM cartridge system £10, Vu-Type £3. Tel. Lancashire 0253 712395.

M128, Cumana single 40/80T drive, Panasonic KX-P1081 printer, all in excellent condition, manuals & books inc. reference manuals, complete 5 yrs BEEBUG magazines, many 5.25" discs £300, also IBM XT PC + software & Epson FX100 printer £100 or £350 the lot. Tel. Bucks 0628 521231.

Binder for BEEBUG, spare brand new £1 to cover p&p. Tel. Hereford 0981 550 344.

Collectors item ?? complete set of BEEBUG magazines Vol.1 Issue 1 to Vol.6 Issue 6, excellent condition, offers? Tel. Southampton 0703 262476.

M128, 15" RGB colour monitor, twin 40/80 Cumana disc drives, Amstrad DMP1 9 dot matrix (tractor), Plug-in ROMs, Viewsheet, Viewstore Database, View Printer Driver, MiniOffice II, Wapping Editor DTP, Advanced Disc Toolkit, NLQ Font Designer, Penpal 2 (lightpen), Quest-paint, Quest mouse, Pace Nightingale Modem, teletext adaptor, user port splitter (all manuals for above), 30 Micro User discs £250. Tel. Surrey 081-337 4247.

A3000, 4Mb RAM, Acorn colour monitor, 80Mb external IDE disc, lots of software including: PipeDream 4, Ovation, Wordz, Squirrel DBM, Base 5, PC Emulator, Hard Disc Companion, Lemmings, dozens of RU discs and magazines, £750, must collect. Tel. Beckenham Kent 081-650 9960 eves.

Local primary school and I are both upgrading - help us pay for it. The following have all been serviced recently: Master 128 £120, BBC B with DFS, ROM/RAM board, ROMs £65, BBC B with DFS £50, 40/80T DS drive, 40T DS drive with PSU £20 each, Cub colour monitor £30, collect or p&p extra. Tel. Suffolk 0394 385799.

Switchbox and cables, BBC to 2 printers £10, Mini Office II, 40T £5, 50x5.25" discs 10p each, Advanced User Guide £5, 30Hr Basic £5, Twin 5.25" disc drives £15, all plus postage. Tel. Barnsley 0226 762450.

Archimedes 8Mb RAM, 103Mb Hard Disc (16 bit SCSI), Aleph 1 30MHz ARM3, RISC OS 3, Laser Direct LBP-8, Spectra Colour Flatbed Scanner, Watford hand held scanner, Impression 2.17 (inc. business supplement), Poster, PC Emulator, OCR, Schema, Autosketch2, Logistix, FontFX, Artisan, loads of utilities, games and over 70 Font Foundary fonts, cost well over £5000 new, bargain at £1,000 (no offers -but may split). Tel. Kent 0892 503786 (work) or 0892 533686 (eves) for further details.

WANTED: Phantom Combat for the BBC (disc preferred), also Colossus 4 Chess (BBC disc only). **FOR SALE:** Morley teletext adaptor +ROM and disc, no reasonable offer refused. Tel. 061-678 2032.

POSTBAG

BACK TO BASICS

If it is not too late, I would like to make a suggestion for an article about Basic before BEEBUG ceases. Ideally I should like something that includes a history of its development, but more realistically what I have in mind is a survey and comparison of what is now available. We have often been told how good BBC Basic is compared with other versions, and I do not doubt it but I do not know where the superiority lies. My own concerns are chiefly with calculation and tabulation of results, so I have been familiar with the ###.## convention of other Basics, and think I have mastered the BBC's @%=& etc.

Two points underlie this suggestion. Only recently have I learnt that there are some Basics that do not use line numbers, and, more relevantly to RISC Developments and Acorn, that Archimedes Basic has an ENDIF. Although I have no intention of changing from my Master, improvement in Basic is more persuasive to me to change than any other feature that the Archimedes offers (apart from increased memory). My academic associates use IBM-compatibles, and I know that on these, though I would have to learn a different Basic, I can get double-precision Fortran that is adequate for any problem I have tackled on a mainframe. Can the Archimedes offer the same? I should be happy to be persuaded to stay with Acorn.

D.Ambrose

The history of Basic is no doubt fascinating, but sadly I do not feel we shall have time to cover it in BEEBUG. To my knowledge it was first defined in the USA by Kemmeny and Kurtz in 1965, and the name is actually an acronym for "Beginners All-purpose Symbolic Instruction Code". BBC Basic has always been a very rich dialect of the language reflecting the open design of the BBC micro. Where other Basics were limited to GOTO and RETURN, BBC Basic allowed procedures and functions to be called by name, a REPEAT-UNTIL statement, and a more comprehensive IF-THEN-ELSE construction than most.

BBC Basic on the Archimedes has removed most of the remaining constraints, with full multi-line IF-THEN-ELSE-ENDIF constructions, a WHILE-ENDWHILE, a CASE statement and more. In addition, the latest version of BBC Basic (Basic VI) now implements IEEE standard 754 using 8-byte representation for real numbers instead of the previous 5-byte representation. A Fortran compiler was available for the Archimedes from Acorn (now no longer available direct), and there is an Archimedes Fortran support group. What more could anyone ask?

PRINTER DRIVER UPDATED

On my early version of View(2.1), one problem is that the printer is initialised at the start of each page when using the SHEETS command for individual sheets of paper, thus losing all the embedded print comments already entered. This problem can be avoided by amending David Holton's excellent printer driver program, published in BEEBUG Vol.12 No.4, as follows:

```
370 LDA #ASC"<": JMP osprint
```

This just gives a carriage return, and therefore preserves the codes.

Alistair Scott

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

RENEWAL RATES FOR BEEBUG MAGAZINE AND MAGAZINE DISC SUBSCRIPTIONS

See May 1993 Editorial for further explanation

The table below shows the renewal rate applying after the June issue 1993 according to the first issue of the renewal period. For joint BEEBUG/RISC User subscriptions add half the appropriate BEEBUG renewal rate to the full RISC User renewal rate; (UK £18.40, Europe & Eire £27.50, Middle East £33.50, Americas & Africa £36.50, Elsewhere £39.50).

Renewal Issue	Issues to go	Mag UK	Mag Europe	Mag Mid-E	Mag Am+Af	Mag Else	Disc UK	Disc O'Seas
Jun	9	16.56	24.75	30.15	32.85	35.55	45.00	50.40
Jul	8	14.72	22.00	26.80	29.20	31.60	40.00	44.80
A/S	7	12.88	19.25	23.45	25.55	27.65	35.00	39.20
Oct	6	11.04	16.50	20.10	21.90	23.70	30.00	33.60
Nov	5	9.20	13.75	16.75	18.25	19.75	25.00	28.00
Dec	4	7.36	11.00	13.40	14.60	15.80	20.00	22.40
J/F '94	3	5.52	8.25	10.05	10.95	11.85	15.00	16.80
Mar '94	2	3.68	5.50	6.70	7.30	7.90	10.00	11.20
Apr '94	1	1.84	2.75	3.35	3.65	3.95	5.00	5.60

BACK ISSUE PRICES (per issue)

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. There is no VAT on magazines.

Volume	Magazine	5"Disc	3.5"Disc
6	£1.00	£3.00	£3.00
7	£1.10	£3.50	£3.50
8	£1.30	£4.00	£4.00
9	£1.60	£4.00	£4.00
10	£1.60	£4.75	£4.75
11	£1.90	£4.75	£4.75

POST AND PACKING

Magazines and discs are postcode a
Please add the cost of p&p when ordering.
When ordering several items use the highest price code, plus half the price of each subsequent code. UK maximum £8.

Post Code	UK, BFPO Ch.I	Europe, Eire	Americas,Africa, Mid East	Elsewhere
a	£1.00	£1.60	£2.40	£2.60
b	£2.00	£3.00	£5.00	£5.50

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 840303, FAX: (0727) 860263
Office hours: 9am-5pm Mon-Fri Showroom hours: 9am-5pm Monday to Saturday
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by RISC Developments Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Editorial Assistance: Marshal Anderson
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Subscriptions: Helen O'Sullivan
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc in machine readable form using plain text format if possible for text, but please ensure an adequate written description is also included of your submission and the contents/format of your disc.

In all communication, please quote your membership number.

Printed by Arlon Printers (0923) 268328 ISSN - 0263 - 7561

Magazine Disc

DECEMBER 1993

BEEBART - The magazine disc contains two separate versions of this comprehensive art package (for different modes) both selectable through a front-end menu.

M-BASE - Part two of this comprehensive database program has now been added to last month's installment to provide the complete application to date.

PUTTING ADFS DIRECTORIES TO WORK - The disc contains the program *Contents* which compiles an easy-to-use index/menu to the contents of any ADFS format disc.

PYRAMID PATIENCE - To while away all those hours of spare time at Christmas and the New Year we have provided this marvellous patience game to keep you entertained.

BEEBUG WORKSHOP - The magazine disc contains a file with the routines listed in the magazine together with a complete working demo of the Quick Sort.

FIRST COURSE - This program, described in the magazine, illustrates the use of the file handling instructions described in this month's article in this series.

MR TOAD'S MACHINE CODE CORNER - This demonstration program shows how to package up a Basic program so that it can be called with a star command from sideways RAM.

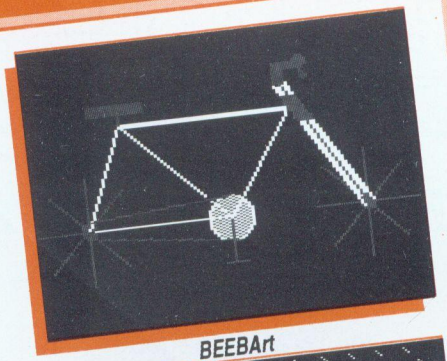
RELOCATOR - This is a useful utility which will modify the addresses of suitable machine code programs so that they can be relocated in memory.

MAGSCAN DATA - Bibliography for this issue of BEEBUG (Vol.12 No.7).

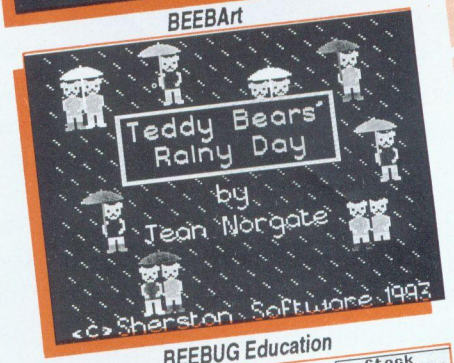
BONUS ITEMS

ROBOL - We have included on this disc an updated version of the popular Robol game which now allows you to retract any unsatisfactory move, rather than start all over again.

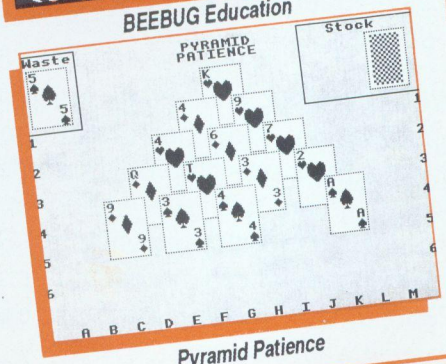
PERPETUAL CALENDAR - Just right for the new year, this updated program is an ideal way of displaying or printing monthly calendars for any year (on the Panasonic KX-P1124 and similar Epson compatible printers).



BEEBArt



BEEBUG Education



Pyramid Patience

ALL THIS FOR £4.75 (5.25" & 3.5" DISC) + £1 P&P (50p FOR EACH ADDITIONAL ITEM)

Back issues (5.25" and 3.5" discs from Vol.6 No.1) available at the same prices.

FOR DISC (5.25" or 3.5") RENEWAL SUBSCRIPTION RATES (UK AND OVERSEAS) PLEASE SEE TABLE ON FACING PAGE

Prices are inclusive of VAT and postage as applicable. Sterling only please.

RISC Developments, 117 Hatfield Road, St.Albans, Herts AL1 4JS

STOP PRESS . . . STOP PRESS . . . STOP PRESS . . . STOP PRESS . . . STOP PRESS . . .

BEEBUG

Special Offer

A5000 Computers for only £850

(Prior to Sept'93 the same system was £1285)

Once again we have managed to obtain a number of A5000 Learning Curve systems at a ridiculously low price. On the previous occasion we sold all 100 in only **ten days!** So hurry if you are not to miss out this time.

The A5000 on offer is the top of the range computer from Acorn, and features the 25MHz ARM3 processor (the latest models are equipped with a 33MHz ARM3 making them marginally faster).

We are offering a range of specially priced upgrades provided they are all supplied or fitted at the same time of your order for the A5000. All are fully guaranteed for 12 months along with the computer. So whether you can only afford the computer now at £850, or a full blown system we can supply exactly what you need.

A5000 Specification
RISC OS 3.1
2Mb RAM
80Mb Hard drive
25MHz ARM3

Learning Curve pack
PC Emulator v1.8
DOS 5
First Word Plus
Acorn DTP
Audio Training Tape
Pacmania
Genesis Plus

Monitor options:

Standard (AKF40)	£149
MultiScan (AKF18)	£179

RAM Upgrades:

2-4Mb	£85
2-8Mb	£379

Additional Hard Drives:

160Mb	£199
260Mb	£279
450Mb	£449

These drives are fitted in addition to the standard 80Mb and do not replace them.

BEEBUG Ltd 117 Hatfield Road, St Albans, Hertfordshire AL1 4JS.
Tel: 0727 840303 Telesales Direct: 0727 840305 Fax: 0727 860263
Prices shown are exclusive of £8.00 for carriage and VAT.