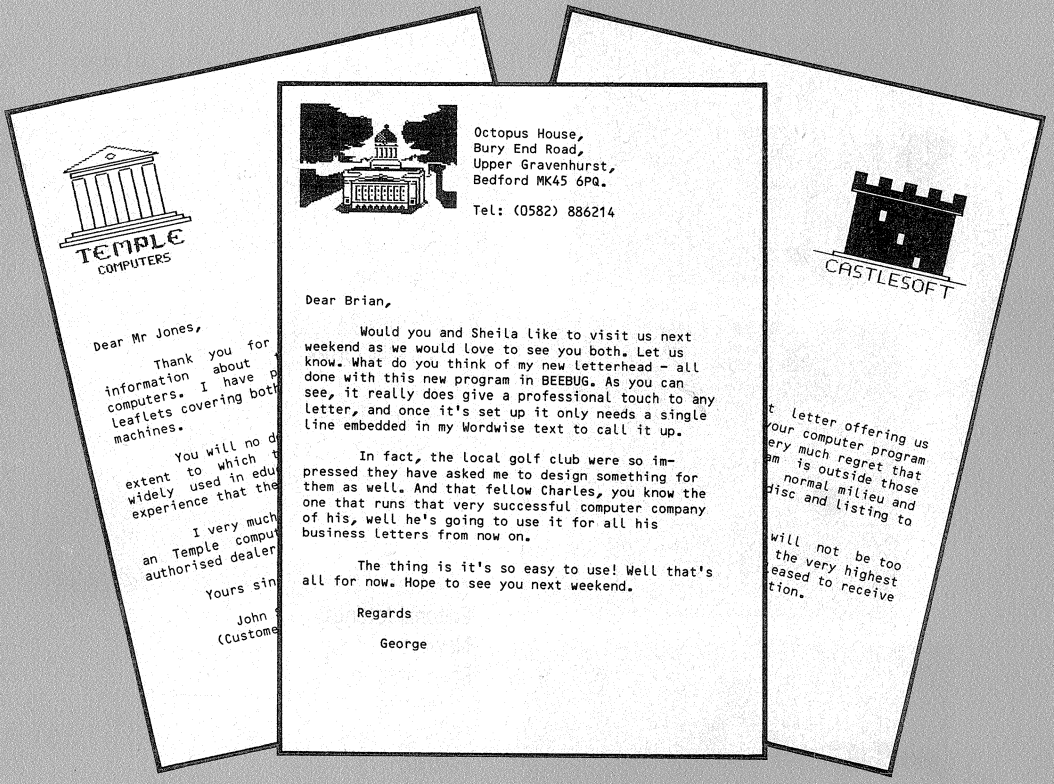
VOLUME 5 ISSUE 8 JAN/FEB 1987 PRICE £1.30

BEEBUG

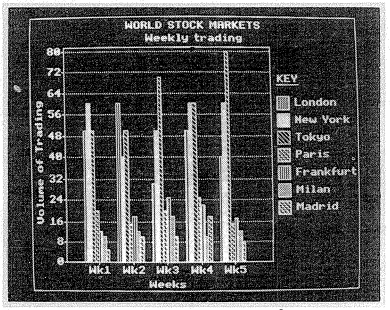
FOR THE BBC MICRO



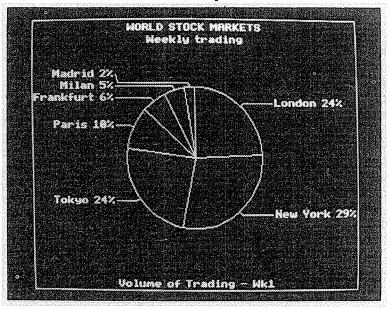
personalized letterheads

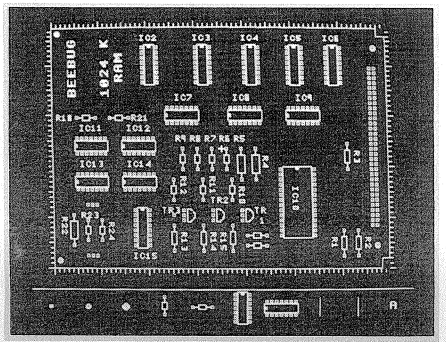
Sideways RAM Buffers

Letter Headings



Business Graphics





PCB Designer

BEEBUG

Volume 5 Number 8
January/February 1987

FEATURES

REVIEWS

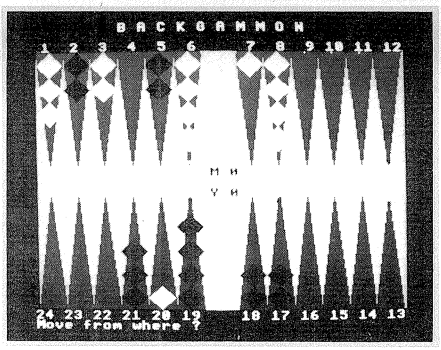
REGULAR ITEMS

HINTS AND TIPS

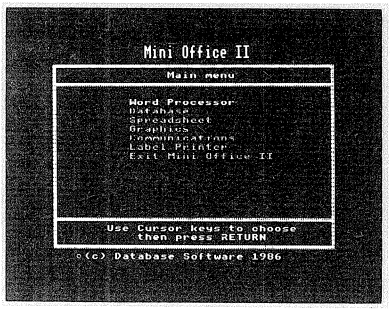
GENERAL

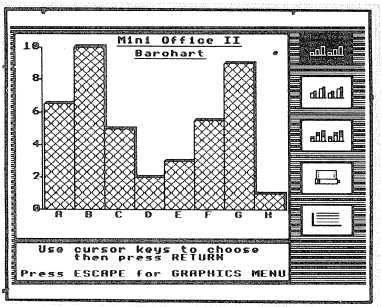
MASTER

Backgammon

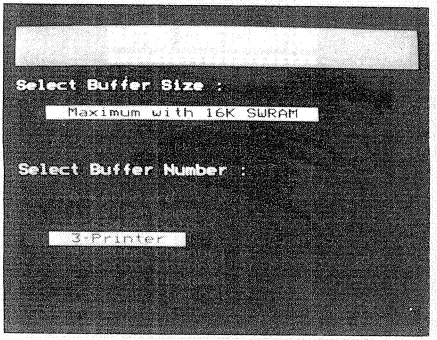


Instant Mini Office





Instant Mini Office



Sideways RAM Buffers

EDITORIAL JOTTINGS

BEEBUG MAGAZINE CHANGES

The changes that we have made to recent issues of BEEBUG, the increase in the number of Master pages and the renumbering of the supplement, have provoked comment from some members. Clearly it is difficult to please everyone, but we can try.

After some thought we have decided to make the following further changes with effect from this issue. The supplement will be reduced to a total of eight pages and will increasingly concentrate on information for members, including our free personal ads section which has proved very popular. This size of supplement will mean that it can be readily removed by those readers who wish to do so, leaving the rest of the magazine intact.

NEW FEATURES

The changes to the supplement provide us with extra editorial pages. We are taking advantage of this to introduce two new and regular features to the magazine covering Communications and Education. These will appear in alternate issues. The first of these new features, entitled The Comms Spot, appears in this issue and provides exciting news of a major new viewdata service.

NEXT ISSUE

This, the January/February issue of BEEBUG, is one of our two double-month issues. However, we shall be endeavouring to bring publication forward, and the March issue should be with you by the end of February.

BEEBUG ON MICRONET

We are currently setting up a substantial data base which will form part of Micronet. Not all the details have yet been finalised but the BEEBUG pages will contain a variety of information, including details of BEEBUG products, and response frames to allow you to join BEEBUG or renew your membership. We expect to be able to release full details next month including the page number of the BEEBUG front page. The BEEBUG pages will also be routed from other pages in Micronet.

CLASSIFICATION

We hope that the new classification symbols for programs and reviews clarify matters with regard to the variety of Acorn systems. The complete set of icons is shown below. These show clearly the valid combinations of machine (version of Basic) and filing system for each item, and Tube compatibility. A single line through a symbol indicates partial working (normally just a few changes will be needed); a cross shows total incompatibility. Reviews do not distinguish between Basic I and II.

COMPUTER SYSTEM

Master (Basic IV)

Compact (Basic IV)

Compact (Basic VI)

Model B (Basic II)

Model B (Basic I)

Electron

FILING SYSTEM

ADFS

DFS

Cassette

TUBE COMPATIBILITY

Tube

News News News News News News News N

Pagemaker Pics

Micro Studio has a range of nine discs, each of 30 pictures, for inclusion in Pagemaker documents. The images are divided into categories. These include computing, sport, people, children, animals, plants, vehicles, objects and a miscellaneous disc with a selection of pictures from all the others.

Each disc costs £2.50, or £2 each for five or more. The discs, or sample printouts, are available from Micro Studio, 83 Clay Street, Soham, Cambs CB7 5HL.

Latest Developments

The machine code program development system from Mijas Software enables software to be easily written for the 6502 (as used in the model B) and 65C02 (used in the Master series).

The system includes a fast macro assembler and a library of machine code routines for inclusion in your own programs under development. The development system costs £22 on disc or £29.50 on ROM from Mijas Software on (0962) 89352.

More Master ROMs

The Overlay board from Vine Micros gives Master 128 owners more ROM sockets. The board plugs into the 1Mbit ROM socket and the ROM plugs into the board. Up to three of the Mega ROM's eight ROM programs can be disabled and substitute 'normal' 16K ROMs plugged into the Overlay board. Extra ROMs can be added to the Master without using any sideways RAM banks or ROM sockets.

The Overlay board costs £19.95. It is only availabe for Master 128s with a socketed Mega ROM. Masters with ROMs soldered in place cannot be fitted with the board. Further details from Vine Micros on (0304) 812276.

Schools Out

The BBC's Telesoftware service starts to broadcast tutorial articles as well as programs this month (January). The new series, called OSBITS, aims to teach BBC micro users the ins and outs of assembler programming and includes both text and example programs.

The broadcasts start in the second week of January and continue, changing weekly, for 26 weeks. All Telesoftware can be viewed and/or downloaded for free by Beeb users with a Teletext adaptor. The Telesoftware pages start at page 700 on BBC2 Ceefax. Further details from BBC Telesoftware (01) 743 5588.

Superior Games

More games have appeared under the Superior Software/Acornsoft label. Elite has been re-released in four versions especially for the Model B, Electron, Master 128/Compact and the 6502 second processor.

Revs is now available complete with the Revs 4-tracks extra race track package and has been altered to include computer aided steering to help novice drivers.

Details from Superior Software on (0532) 459453.

Dirty Den, Lofty & So On

A game based on the TV soap opera Eastenders has been released by licence supremos Macsen Software. The game appeals to both arcade and adventure addicts but is still a 'unique game' claims Macsen. The Eastenders game is supplied on tape and costs £9.95 from Macsen Software on (0267) 232508.

Pearl of a Logger

Massodax have announced a new data logger for the BBC and Master Series. The PEARL 8000 is an 8 channel Parallel Entry Auto Ranging Logger (PEARL).

The unit provides an 8 channel multiplexed input to one of the four analogue ports on the host computer.

The unit measures 10x3x8 inches and is powered from the micro. An introductory price of £150.00 has been set. For details, contact Massodax on Tring (044282) 3091.

Instant Mini Office

When Database Publications launched the ROM version of their highly successful Mini Office, it seemed a good time to take a long hard look at what this system has to offer. Simon Williams reports.

Product: Mini Office II
Supplier: Database Publications
Europa House, 68 Chester Road,
Hazel Grove, Stockport SK7 5NY.
(061) 429 7931
Price: £59.95 inc. VAT

Mini Office has become one of the most successful business packages of all time for the BBC Micro. This is partly due to the copious advertising and reviews it has received in Database's own publications, but also, it has to be said, for the extraordinary value the package offers. Database has now parcelled the six programs into four 32K ROMs, mounted them on a carrier board and re-released them as 'Instant' Mini Office II. In the process, they've added a few extras to make the suite still more attractive.

Mini Office II consists of six modules, covering the main 'serious' applications of the BBC micro in business:

* Word Processor
* Database
* Spreadsheet
* Communications
* Business Graphics
* Label printing

The new packaging of the complete system, a total of 128K of machine-code, means that any of the applications are instantly 'on-tap' with a simple star command.

INSTALLATION

The four ROMs are supplied on a well-designed but tackily made 3" x 5" circuit board, which also holds a few extra switching chips and a header plug connected to a short length of ribbon cable. The other end of the cable is fitted with another header plug, and it's this you insert into any available ROM socket. The complete set of ROMs are mapped onto just one ROM slot.

The carrier board will work happily from one of the sockets of a ROM expansion card, but you will have to be careful where you position the board so that you can still replace the cover of your beeb.

Once it's all installed, you can call up the Mini Office II main menu with the command *MINI, or any of the individual functions with an abbreviation such as *WP for the word processor.

THE WORD PROCESSOR

When you select the word processor, a secondary menu is displayed with a number of filing options. These include load, save, clear, print, preview and edit text, search and replace, as well as screen mode selection, and a facility to use the AMX mouse instead of the cursor keys.

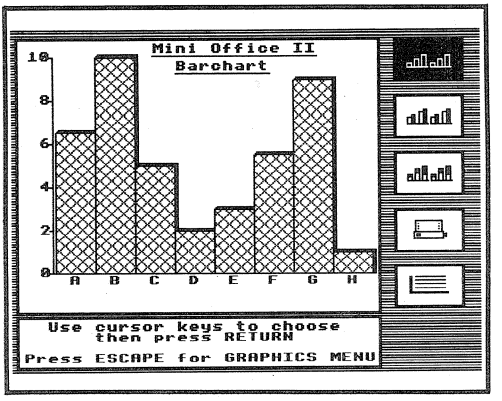
All this is very convenient and brings to mind the opening menu of Wordwise, but with more options. In fact there is a strong thread of Wordwise running throughout the Mini Office word processor, and anyone used to Wordwise will feel very much at home. Some of the controls and the presentation of text have been changed, but most of the alterations are cosmetic (embedded codes being highlighted in magenta rather than green, for instance).

The embedded commands include specific Epson printing effects like double-strike and italics, but others can also be set by using a general Escape code command. You can also specify single or double-sized text and can control height and width

independently to produce a range of special effects. Up to ten sequences of printer controls can be stored separately within Mini Office and called with a single command.

The Wordwise Plus programming language is missing from Mini Office, of course, and there are a few other features where the Computer Concept's product has the edge. When you need to scroll the display, for instance, the Mini Office editing screen moves rather more slowly than Wordwise does. By and large, the Mini Office word processor is extremely good, and has a number of significant advantages over the likes of Wordwise and View.

For a start, unlike Wordwise, Mini Office can display text in 20, 40 or 80 column modes (and use Watford, Aries or Ramamp shadow RAM boards to provide the space for long text files). In the 20 column mode text appears double height as well as double width. In addition to an on-screen word counter and elapsed time clock, you can call up a typing speed meter, which is handy if you like to keep your index fingers in trim.



You can select most of the major page format parameters from a separate menu at print-time, which is a lot more convenient than having to rely completely on embedded commands. And you can combine the facilities of the word processor with the database to process mailing lists.

The manual is clear and comprehensive, but it would have been nice to have had a quick reference card of the available commands as well as a key strip.

THE DATABASE

This module is not as comprehensive as the word processor. Perhaps its main limitation is that you can't define the record format. You can specify the length and type of each field, but they are all listed in order down the screen and there's no way of grouping them together. This may not worry you, but if you have a variety of information on the same record, you will probably want to separate the main areas.

Mini Office is a fixed format disc-based, random-access file manager. It allows around 20 fields per record, although each field can be up to 60 characters long. The program tries to work in memory for as long as possible, so keeping access times down.

The Mini Office database can hold formulae to calculate values from numeric fields on any record and deposit the results in other named fields. It can also total fields across each record in a file.

The database can create subsets of its records according to a set rule which you define. The rule can include the logical operators =, <>, >, <, >=, <= and INSTR. This last operator will find an occurrence of a given string anywhere within a specified field. You can search on any number of fields. If the file is small enough to be held in memory, searching is very fast, although it obviously slows down when the file becomes disc-bound.

The data can be sorted on one or more fields and in ascending or descending order. You can print out all or selected fields from within the database, and combine data with text from the word processor to produce more sophisticated print-outs.

THE SPREADSHEET

Like the word processor, this module works in any of the BBC micro's screen modes, and can make use of shadow RAM. It is an intelligent spreadsheet in that it can distinguish automatically between text and number entries. You need only use a special key (f0 in this case) to precede formulae.

You can define a bigger sheet in the ROM version of Mini Office than in the disc or cassette versions of the suite,

anything up to 5148 cells. The shape of the sheet is user-definable and this includes individual or global definition of column widths.

The mathematical operators include +, -, *, /, total, maximum and minimum, and an extended set of trigonometric and logarithmic operators has been added for the ROM version. It's disappointing not to see statistical functions which are perhaps of more use in the business sphere, but to some extent these can be built up from the existing functions.

The spreadsheet normally re-calculates automatically, and impressively quickly. It offers all the normal formula replication facilities, except replicating a formula into a range of cells. Each one has to be copied individually.

There seems little in the 'standard' BBC Micro spreadsheets like Viewsheet or Ultracalc that Mini Office can't reproduce. It's perhaps marginally slower in scrolling around the sheet, but this isn't really a problem. The menu lets you save data from a sheet in a suitable form for the graphics module to plot.

THE GRAPHICS MODULE

This part of the suite is intended to produce graphs and charts from figures in the spreadsheet, or from data typed at the keyboard. It's the only part of Mini Office which makes any attempt to use icons for its menu options, although all modules will work with the AMX mouse.

The graphics module produces bar charts, pie charts and line graphs. Bars can be 2D or 3D, separate or stacked and pies can be exploded and with default shading patterns (which didn't display with a Ramamp shadow RAM board on) or those selected with the keyboard or mouse. You can scale the graphs yourself or have Mini Office do it for you. Custom-made titles can also be added.

Finished graphs can be displayed on screen or sent to any Epson-compatible printer. I've never been very convinced of the usefulness of business graphics, but this program is probably as good as any.

THE COMMS MODULE

This program is designed to let you communicate with information services like Prestel or Telecom Gold. It provides the main communication protocols in a convenient menu, and in most cases you can simply select one of three main options.

You can also define your own protocols, but this takes you down some tortuous menu paths and is not nearly as easy to use as a program like CommStar. In use, the comms module proved quite adequate and it can cope with the extended ASCII character set (assuming your correspondent can also). This is handy if you want to send files where the high bit of each byte is used (tokenised Basic files, for instance). Overall, the comms module is usable, but not as convenient as some stand-alone packages.

THE LABEL PRINTER

This is a comprehensive utility for printing labels up to three across on an 80 column printer. You can select the number of lines, line gap and three tab positions to line up the start of text. You can save any label format you define, and import data from the database to print labels from a datafile.

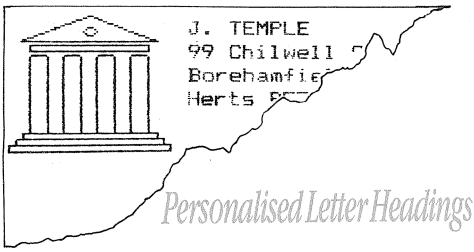
VERDICT

The first conclusion that must be drawn is that Instant Mini Office II is exceptional value for money. To have put six high quality programs into one package and have marketed the lot for under £60 must be doing the BBC micro user a favour.

Having said that, there are a few shortcomings to the programs, the main one perhaps being that no real attempt has been made to integrate the separate parts of the 'Office'. Although you can merge data from the database with text from the word processor and produce graphics from the spreadsheet, there is no facility to share datafiles as there is, for example, in the Interlink series.

Any criticisms along these lines, though, are likely to look churlish when you remember that the 128K of Mini Office costs less than some single applications of similar quality.

We have been able to arrange a special deal with Database Publications and can offer Instant Mini Office II at 10% discount to members, making the price just £53.95 - see Mail Order catalogue.



For all those who are addicted to using their micros for correspondence, Johnathan Temple has produced a utility to adorn your letters with a stylish logo or design of your choice.

Now that so many printers are capable of both good quality text and graphics, what better than to mix the two together to produce sophisticated and attractive letterheads for both personal and business use. The program presented here allows you to do just that - using it you can easily design your own logos, which can then be printed out and used in letterheads etc. as you wish. This can be done quite automatically from within Wordwise.

TYPING IN THE LOGO DESIGNER

The main program is essentially a logo designer. To use this program you will need a printer that supports Epson-compatible graphics; very few don't. Type the program in and then save it on cassette or disc. Now type:

PAGE=&1400 <Return>

and CHAIN the program back in. This is done to remove the problem of downloading for DFS users, but tape users must also re-load it. Master and Compact users need not worry about resetting PAGE at all. Please note that model B and B+ users with the ADFS fitted will not be able to use the logo designer with this filing system, though logos can subsequently be printed from ADFS files.

While testing the program, it is worth noting that f0 can be used at any time to exit from the program, as the Escape key is effectively disabled. The program is automatically recovered should you press Break, but after Ctrl-Break, you will need to re-enter PAGE=&1400<Return> followed by OLD<Return> to recover.

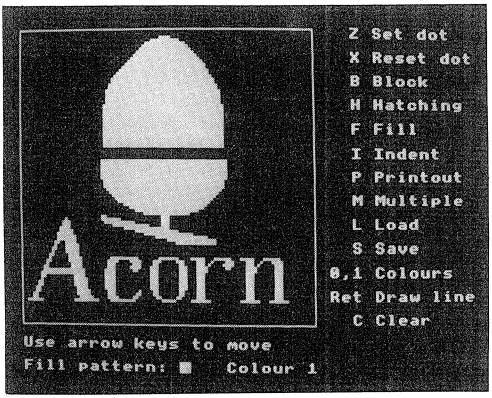
Finally, if you are worried about typing the program in (or wish to avoid sore fingers) then a full version of the program is included on this month's magazine cassette/disc.

USING THE LOGO DESIGNER

The logo designer has similar features to many drawing packages, the main difference to these being that the design area is made up of less dots or pixels. For clarity, these dots are shown enlarged. When you run the program, you will see a large outline box, to the right of which is a list of all the operations. Below the box, the current colour and fill pattern will be displayed.

At the bottom left of the design area (the outlined box) is the cursor, a small red block. This can be moved across the area using the cursor keys; pressing Shift as well will speed it up.

The Z key will set a dot at the current cursor position, and X will erase it. When creating large logos, a very useful feature is line drawing - pressing Return will draw a line from the last point plotted to the current cursor position. Pressing B will give a large block at the current cursor position.



Also available is a simple fill; pressing F will fill an enclosed shape in the present pattern, which is displayed below the design area. The fill is reasonably quick considering the amount of

work involved, but the penalty for this is that complex shapes may need several fills for different sections.

The fill pattern can be changed by pressing H to switch 'hatching' (a dotted effect) on or off. Using the Ø and 1 keys to switch between black or white will also affect the pattern. The current 'colour' will also be used in line drawing and the block operation.

If you're not happy with your design, then the quickest way to get rid of it is to clear the grid by pressing C. Another useful feature for correcting mistakes is that by setting the colour to Ø the B key can act as an eraser, rubbing out large blocks. Unsatisfactory lines can be easily removed by drawing the line again but in colour Ø.

Once you've come up with a design that you're happy with, or wish to safeguard your hard work from power cuts etc., then press S. After entering a suitable filename (not more than seven letters with the DFS) your logo will be saved to tape or disc. Note that all logos are saved with the prefix "H." - this is done so that disc users will find all their logos in directory H, to avoid confusion. ADFS users will need to create such a directory first (e.g. use *CDIR $.H).

To load a logo, for adjustments or printing out, use the L key. The filename should be entered again, and the logo will then be loaded; this takes a while, so please be patient!

Of course, you'll want to be able to print out your final logo design. Just press P to do this. M will perform basically the same function, but will print as many copies as you like, 'form-feeding' between each one.

Usually the logo is printed at the left-hand edge of the paper, but this can be changed using I to set an indent of between Ø and 63 spaces.

The Escape key can generally be used to abort an operation, which can be very useful should any function (especially the fill) start going wrong, or you press the wrong key. However, Escape cannot be used to interrupt a filing operation (load or save) once the filename has been entered.

COMMANDS

Z	Turn pixel on	L	Load design
X	Turn pixel off	S	Save design
B	Large block	Ø/1	Change colour
H	Toggle hatching	Ret	Draw line
F	Fill shape	C	Clear grid
P	Print single copy	I	Indent logo
M	Print multiple copies		

USING YOUR LOGOS

The main use for your logos will probably be as part of a letterhead. To do this, you will need to use P or M from within the designer to print out the logo, and then turn off the printer and wind the paper back in ready for printing a letter around and/or below it.

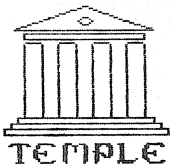
```
OC27,69

LM17 J. TEMPLE
99 Chilwell St,
Borehamfield,
Herts BE3 8UG.
```

figure 1

For Wordwise disc users, the whole process can be completely automated using a special 'converter' program (Listing 2). What this does is to allow logos to be printed from within a Wordwise document using the little-known embedded command:

GF"<filename>"

which will print out the relevant file. For printers with reverse feed capability, this will be used so that the print head returns to the top of the page ready for an address etc.



J. TEMPLE
99 Chilwell St,
Borehamfield,
Herts BE3 8UG.

figure 2

If you wish to use the GF facility, type in the Converter program and save it. Now run the program and enter the filename of the saved logo you wish to convert to Wordwise format. Once this has been done, the program will produce a converted file with the same name as the previous one, but with the prefix "W."

This filename can then be used with the GF embedded command from Wordwise. The commands for an example letterhead are shown in figure 1, and the result in figure 2.

```
  10 REM Program LOGO DESIGNER
  20 REM Version B3.3
  30 REM Author  Jonathan Temple
  40 REM Beebug  Jan/Feb 1987
  50 REM Program subject to copyright
  60 :
 100 MODE 1
 110 !&70=!&1900
 120 HIMEM=&2B00
 130 I%=&2B00:D%=&2B01
 140 PROCinit
 150 PROCclear
 160 ON ERROR PROCerror
 170 PROCedit
 180 MODE 7
 190 *FX 4
 200 *FX 21
 210 *FX 200
 220 END
 230 :
1000 DEFPROCedit
1010 PROCoff
1020 X%=0:Y%=0:PROCstatus
1030 REPEAT
1040 PROCon
1050 J%=0:K%=0
1060 IF INKEY-26 J%=-8
1070 IF INKEY-122 J%=8
1080 IF INKEY-58 K%=8
1090 IF INKEY-42 K%=-8
1100 IF INKEY-1 J%=J%*4:K%=K%*4
1110 IF J% OR K% PROCmove
1120 IF INKEY-67 PROCplot(0)
1130 IF INKEY-98 PROCplot(1):PROCst
1140 IF INKEY-40 G%=0:PROCstatus
1150 IF INKEY-49 G%=1:PROCstatus
1160 IF INKEY-85 H%=H% EOR1:PROCstatus
1170 IF INKEY-83 PROCclear
1180 IF INKEY-74 PROCline
1190 IF INKEY-101 PROCblock
1200 IF INKEY-68 PROCfill
1210 IF INKEY-38 PROCindent
1220 IF INKEY-56 PROCprintout(1)
1230 IF INKEY-102 PROCprintout(2)
1240 IF INKEY-87 PROCfile("LOAD"):PROCd
isplay:*FX 200
1250 IF INKEY-82 PROCfile("SAVE"):PROCs
tatus:*FX 200
1260 UNTIL INKEY-33
1270 ENDPROC
1280 :
1290 DEFPROCmove
1300 PROCoff
1310 X%=X%+J%:Y%=Y%+K%
1320 IF X%<0 THEN X%=792+X%
1330 IF Y%<0 THEN Y%=768+Y%
1340 IF X%>784 THEN X%=X%-792
1350 IF Y%>760 THEN Y%=Y%-768
1360 PROCon
1370 ENDPROC
1380 :
1390 DEFPROCline
1400 J%=X%/8:K%=Y%/8
1410 PROCwait
1420 PROCplot(G%):j%=x%-J%:k%=y%-K%
1430 L%=SQR((J%-x%)^2+(K%-y%)^2)
1440 IF L%=0 THEN ENDPROC
1450 FOR N%=1 TO L%
1460 X%=8*INT(J%+N%*j%/L%):Y%=8*INT(K%+
N%*k%/L%)
1470 PROCplot(G%):NEXT
1480 x%=J%:y%=K%
1490 X%=J%*8:Y%=K%*8:PROCstatus
1500 ENDPROC
1510 :
1520 DEFPROCblock
1530 IF X%>720 OR Y%<64 VDU 7:ENDPROC
1540 j%=X%/8:k%=Y%/8
1550 PROCoff
1560 FOR K%=k%-7 TO k%
1570 M%=D%+(K%DIV8)*99:F%=2^(K% AND7)
1580 FOR J%=j% TO j%+7
1590 IF G% M%?J%=(M%?J% OR F%) ELSE M%?
J%=(M%?J% AND 255-F%)
1600 NEXT,
1610 GCOL 0,G%:MOVE X%,Y%+4
1620 PLOT 0,60,0:PLOT 81,0,-60
1630 PLOT 0,-60,0:PLOT 81,0,60
1640 PROCon
1650 ENDPROC
1660 :
1670 DEFPROCfill
1680 C%=G%:J%=X%:K%=Y%
1690 PROCwait
1700 FOR L%=-8 TO 8 STEP 16
1710 Y%=K%:REPEAT:C%=G%
1720 IF X% MOD16=Y% MOD16 C%=G% EOR H%
1730 REPEAT
1740 PROCplot(C%):C%=C% EORH%:X%=X%+8
1750 UNTIL FNread=G% OR X%>784
1760 X%=J%:C%=G%
1770 IF X% MOD16=Y% MOD16 C%=G% EOR H%
1780 REPEAT
1790 PROCplot(C%):C%=C% EORH%:X%=X%-8
1800 UNTIL FNread=G% OR X%<0
1810 X%=J%:Y%=Y%+L%
1820 UNTIL FNread=G% OR Y%<0 OR Y%>760
1830 NEXT
1840 X%=J%:Y%=K%:PROCstatus
1850 ENDPROC
1860 :
1870 DEFPROCindent
1880 REPEAT *FX 21
1890 PRINT TAB(0,28);SPC(39)
```

```
 1900 INPUT TAB(0,28);"Indent spaces (0-
63) : " z%
 1910 UNTIL z%>=0 AND z%<64
 1920 ?I%=z%:PROCstatus
 1930 ENDPROC
 1940 :
 1950 DEFPROCprintout(N%)
 1960 PROCwait
 1970 IF N%=2 N%=FNmultiple
 1980 FOR C%=1 TO N%
 1990 VDU 2,1,27,1,65,1,8
 2000 FOR K%=11 TO 0 STEP -1
 2010 IF ?I%>0 FOR L%=1 TO ?I%:VDU 1,32:
NEXT
 2020 VDU 1,27,1,76,1,198,1,0
 2030 M%=D%+K%*99
 2040 FOR J%=0 TO 98
 2050 VDU 1,M%?J%,1,M%?J%:NEXT
 2060 VDU 1,10:NEXT
 2070 IF N%>1 AND C%<N% VDU 1,12
 2080 NEXT:VDU 1,27,1,65,1,12,3
 2090 PROCstatus
 2100 ENDPROC
 2110 :
 2120 DEFFNmultiple
 2130 REPEAT *FX 21
 2140 PRINT TAB(0,28);SPC(30)
 2150 INPUT TAB(0,28);"Copies: " N%
 2160 UNTIL N%>0
 2170 =N%
 2180 :
 2190 DEFPROCon
 2200 B%=FNread:GCOL 0,2
 2210 PLOT 69,X%,Y%:PLOT 65,4,0
 2220 PLOT 65,0,4:PLOT 65,-4,0
 2230 ENDPROC
 2240 :
 2250 DEFPROCoff
 2260 PROCwrite(B%):GCOL 0,B%
 2270 PLOT 69,X%,Y%:PLOT 65,4,0
 2280 PLOT 65,0,4:PLOT 65,-4,0
 2290 ENDPROC
 2300 :
 2310 DEFPROCplot(C%)
 2320 PROCwrite(C%):GCOL 0,C%
 2330 PLOT 69,X%,Y%:PLOT 65,4,0
 2340 PLOT 65,0,4:PLOT 65,-4,0
 2350 ENDPROC
 2360 :
 2370 DEFPROCst
 2380 x%=X%/8:y%=Y%/8
 2390 ENDPROC
 2400 :
 2410 DEFPROCwrite(N%)
 2420 Z%=Y%/8:W%=(Z%DIV8)*99
 2430 M%=D%+X%/8+W%:F%=2^(Z% AND 7)
 2440 IF N% ?M%=(?M% OR F%) ELSE ?M%=(?M
% AND 255-F%)
 2450 ENDPROC
 2460 :
 2470 DEFFNread
 2480 Z%=Y%/8:W%=(Z%DIV8)*99
 2490 M%=D%+X%/8+W%
 2500 =ABS((?M% AND 2^(Z% AND 7))>0)
 2510 :
 2520 DEFPROCfile(C$)
 2530 PROCoff
 2540 LOCAL X%,Y%
 2550 REPEAT *FX 21
 2560 PRINT TAB(0,28);SPC(30)
 2570 INPUT TAB(0,28);"Filename: " F$
 2580 UNTIL F$>"" AND LEN(F$)<8
 2590 *FX 200,1
 2600 $&700=C$+" H."+F$+" 2B00"
 2610 IF C$="SAVE" $&700=$&700+" 2FFF":C
$="SAV"
 2620 PRINTTAB(0,28)C$;"ING """;F$;""""";
SPC(9)
 2630 X%=0:Y%=&7:CALL &FFF7
 2640 IF C$="LOAD" VDU24,0;0;792;768;16
 2650 ENDPROC
 2660 :
 2670 DEFPROCdisplay
 2680 PRINT TAB(0,28);"Please wait";SPC(
25)
 2690 GCOL 0,1
 2700 FOR K%=0 TO 95
 2710 M%=D%+(K%DIV8)*99:F%=2^(K% AND 7)
 2720 FOR J%=0 TO 98
 2730 IF (M%?J% AND F%) PLOT69,J%*8,K%*8
:PLOT65,4,0:PLOT65,0,4:PLOT65,-4,0
 2740 NEXT,
 2750 PROCstatus
 2760 ENDPROC
 2770 :
 2780 DEFPROCwait
 2790 PROCoff
 2800 PRINT TAB(0,28);"Please wait";SPC(
25)
 2810 ENDPROC
```

```
 2820 :
 2830 DEFPROCdelstat
 2840 PROCoff
 2850 PRINT TAB(0,28);SPC(30)
 2860 ENDPROC
 2870 :
 2880 DEFPROCstatus
 2890 PROCon
 2900 PRINT TAB(0,28);"Use arrow keys to
move";SPC(10)
 2910 PRINT TAB(0,30);"Fill pattern: ";
 2920 COLOUR G%:IF H%<>G% COLOUR129
 2930 VDU 224,17,3,17,128
 2940 PRINT ;"   Colour ";G%
 2950 ENDPROC
 2960 :
 2970 DEFPROCinit
 2980 VDU 23,224,204,204,51,51,204,204,5
1,51
 2990 VDU 23;10,32;0;0;0;
 3000 VDU 29,16;176;
 3010 VDU 19,1,3;0;19,2,1;0;17,3
 3020 MOVE -16,-16
 3030 PLOT 1,824,0:PLOT 1,0,792
 3040 PLOT 1,-824,0:PLOT 1,0,-792
 3050 FOR N%=2 TO 26 STEP 2
 3060 READ L$,A$
 3070 PRINTTAB(30-LEN(L$),N%);L$;" ";A$
 3080 NEXT
 3090 G%=1:H%=0:?I%=0
 3100 X%=0:Y%=0:B%=0
 3110 *KEY 10 !&1900=!&70|M PAGE=&1400|M
OLD|M PRINT'"Type:"'"*SAVE <file> 2B00
2FFF"'"to recover logo"'|L|M
 3120 ENDPROC
 3130 :
 3140 DEFPROCclear
 3150 FOR N%=0 TO 1184 STEP 4
 3160 D%!N%=0:NEXT
 3170 VDU 24,0;0;792;768;16
 3180 x%=0:y%=0:B%=0
 3190 PROCon
 3200 ENDPROC
 3210 :
 3220 DEFPROCerror
 3230 VDU 2,1,27,1,65,1,12,3
 3240 IF ERR=17 THEN ENDPROC
 3250 IF ERR<128 CLS:REPORT:PRINT" at li
ne ";ERL:GOTO 190
 3260 PROCdelstat
 3270 *FX 21
 3280 PRINT TAB(0,27);:REPORT:PRINT" - P
ress a key"
 3290 k%=GET
 3300 PRINT TAB(0,28);SPC(39)
 3310 ENDPROC
 3320 :
 3330 DATA Z,Pixel on,X,Pixel off
 3340 DATA B,Block,H,Hatching
 3350 DATA F,Fill,I,Indent
 3360 DATA P,Printout,M,Multiple
 3370 DATA L,Load,S,Save
 3380 DATA "0,1",Colours
 3390 DATA Ret,Draw line,C,Clear
```

```
   10 REM Program WORDWISE/+ CONVERTER
   20 REM Version B1.2
   30 REM Author  Jonathan Temple
   40 REM Beebug  Jan/Feb 1987
   50 REM Program subject to copyright
   60 :
  100 MODE 6
  110 HIMEM=&4E00
  120 W%=HIMEM:D%=HIMEM+&D00
  130 PROCload
  140 PROCconvert
  150 PROCsave
  160 END
  170 :
 1000 DEFPROCload
 1010 REPEAT
 1020 CLS
 1030 INPUTTAB(0,3) "Filename: " F$
 1040 UNTIL F$>"" AND LEN(F$)<8
 1050 $&700="L. H."+F$+" "+STR$~D%
 1060 X%=0:Y%=&7:CALL &FFF7
 1070 ENDPROC
 1080 :
 1090 DEFPROCconvert
 1100 PRINT'"Please wait"
 1110 P%=W%:I%=?D%
 1120 ?P%=27:P%?1=65:P%?2=8
 1130 P%=P%+3
 1140 FOR K%=11 TO 0 STEP -1
 1150 IF I%>0 FOR L%=0 TO I%-1:P%?L%=32:
NEXT:P%=P%+I%
 1160 ?P%=27:P%?1=76:P%?2=198:P%?3=0
 1170 P%=P%+4
 1180 M%=D%+K%*99+1
 1190 FOR J%=0 TO 98
 1200 ?P%=M%?J%:P%?1=M%?J%
 1210 P%=P%+2:NEXT
 1220 ?P%=10:P%=P%+1:NEXT
 1230 ?P%=27:P%?1=65:P%?2=12
 1240 P%?3=27:P%?4=106:P%?5=255
 1250 P%?6=27:P%?7=106:P%?8=33
 1260 P%=P%+10
 1270 ENDPROC
 1280 :
 1290 DEFPROCsave
 1300 $&700="S. W."+F$+" "+STR$~W%+" "+S
TR$~P%
 1310 X%=0:Y%=&7:CALL &FFF7
 1320 PRINT'"Conversion complete"
 1330 ENDPROC
```

OS Entry Points

Reference information in a compact form can save hours of searching through the growing number of BBC Micro and Master Series reference works. From time to time we will be presenting various tables of useful information for quick reference use. It is hoped that you will find these sufficiently useful to photocopy so as to have them instantly to hand. We start this month with a collection of operating system calls, addresses and vectors. Our major sources for this are the Beebug Reference Card, the BBC Micro Advanced User Guide and the Master Reference Manual Part One.

Entry name	Entry point	Vector name	Vector address	Function
		USERV	200	Reserved for user routines
		BRKV	202	Break vector
		IRQ1V	204	IRQ vector
		IRQ2V	206	Unrecognised IRQ vector
OSCLI	FFF7	CLIV	208	Interpret the command line
OSBYTE	FFF4	BYTEV	20A	Miscellaneous OS operation
OSWORD	FFF1	WORDV	20C	Miscellaneous OS operation
OSWRCH	FFEE	WRCHV	20E	Write character to screen
OSNEWL	FFE7			Write LF CR to screen
OSASCI	FFE3			Write character to screen
OSRDCH	FFE0	RDCHV	210	Read character from keyboard
OSFILE	FFDD	FILEV	212	Read or write complete file
OSARGS	FFDA	ARGSV	214	Read or write data about file
OSBGET	FFD7	BGETV	216	Read single byte from file
OSBPUT	FFD4	BPUTV	218	Write single byte to file
OSGBPB	FFD1	GBPBV	21A	Read/write block of bytes to file
OSFIND	FFCE	FINDV	21C	Open or close a file
GSREAD	FFC5			Read a byte from a string
GSINIT	FFC2			Initialise GSREAD string
OSEVEN	FFBF			Generate an event
OSRDSC	FFB9			Read a byte from screen or ROM (more commonly known as OSRDRM)
OSWRSC*	FFB3			Write a byte to screen
		FSCV	21E	Filing system control vector
		EVENTV	220	Event interrupt vector
		UPTV	222	User print routine vector
		NETV	224	Econet vector
		VDUV	226	Unrecognised VDU command vector
		KEYV	228	Keyboard control vector
		INSV	22A	Buffer insert vector
		REMV	22C	Buffer remove vector
		CNPV	22E	Buffer count/purge vector

* Master series only

Sideways RAM Update

Two of the diagrams published with this article in BEEBUG Vol.5 No.7 may not be as clear as they should be. In Fig.1 the lead connecting the probe is connected ONLY to the 6264 RAM (pin 27, which also has the 10K Ohm resistor attached). In Fig.5, pin 26 of the 2764 should show N.C. - Not Connected - and not A13, while pin 15 should show D3 (not D5).

THE COMMS SPOT

Peter Rochford starts the first of a regular series of bi-monthly reports on the communications scene by revealing the details of a new view-data service soon to be launched nationwide.

This month's issue of BEEBUG sees the launch of our section devoted to communications, an area of fast-growing interest to many micro owners. It will be a bi-monthly feature and naturally be biased towards owners of the BBC B and Master Series computers.

We hope to provide a wide range of features for you, including teach-ins for those new to comms, reviews of new hardware and software, a look at the various dial-up services available, regular gossip and chat from the comms scene, and we'll try to answer questions and queries on any aspect of comms. For the latter we need YOU. So, if you have a particular problem with your comms setup or if there is anything to do with comms you are puzzled about - write to us! We will try to fit as many letters and answers in as we can. Address your letters to The Comms Spot at BEEBUG's St.Albans address and they will be forwarded onto me.

To kick off this new section, we have news of a viewdata service being launched that is claimed to be far more powerful than Prestel and with potentially much wider appeal.

EPNITEX

Those who subscribe to Prestel, BT's own dial-up viewdata service, may already know of an Information Provider called Timefame International. Timefame was one of the first IPs on Prestel and had a large database providing on-line services to many large businesses.

Alongside these business services, Timefame ran its bulletin board section where any Prestel user could fill in a response frame and write his or her opinions on a range of subjects for others to see. The frames were routed in a carousel fashion, and appeared within seconds of being sent. In this way, people from one end of the country to the other could exchange opinions, ideas and information.

All this was free of charge, and the area was very popular on Prestel, giving Timefame the number one slot for highest frame accesses of any IP. Areas on the BB included Letterbox for general topics whilst others were for special interests, including Microboard for computer users. Microboard was VERY popular and used by many Beeb owners.

I have been using the past tense when describing Timefame's activities on Prestel, as they have now quit the system after a disagreement with BT over the conditions of their contract.

Timefame are now to launch their own viewdata service called Epnitex which should be up and running by the time you read this. I hope in next month's magazine to give you a full run down on what it has to offer after I have been allowed to take a look round it.

Meanwhile, after a telephone conversation with Timefame's MD, Roy Norman, I can give some details about what to expect from Epnitex. I should point out that he was being rather guarded about all the features of the new system until its launch. So what follows I am assured, is only a small part of what Epnitex has to offer.

The format for Epnitex will be of the viewdata standard, like Prestel. Therefore those already using Prestel can utilise existing equipment and software to access it. However, Epnitex it is claimed, will have far more powerful and extensive facilities than Prestel. For example, those who use Prestel will know what a chore it can be sometimes to work your way through menu after menu to reach a desired area or frame. Epnitex will feature a very sophisticated word-search system that can find any particular area by entering its name. In fact you don't have to enter all of the name, as just enough to distinguish and identify it will do.

There will be no frame charges on the new system, which I find surprising, and no electronic mail charges either. How IPs are to get revenue without frame charges I do not quite understand.

On the subject of IP's, it is interesting to note that after their row with BT on Prestel, Timefame are at pains to stress that IPs on Epnitex will face far less restrictions on what they can do. This applies to both the content and the updating of an IP's database. On Epnitex, IP's can do their own updating with the very simplest of terminal software and all changes are instant.

One important feature that any IP would welcome is the ability to restrict access to any part of their database without any need to contact Timefame. It can all be done remotely from the IP's own terminal and makes the setting up of closed user groups really very simple and quick indeed.

Timefame say the mailing system, called Epnimail is far more sophisticated than Prestel's mailbox system. Mailbox frames will allow a far larger number of characters to be used and without the loss of room after using ESC codes (Escape codes are an essential part of Prestel because it is based on a 7 bit system - Ed).

Extensive wordprocessing facilities will be available when entering text, allowing for example, insert/overwrite and wordwrap. Wordwrap is something Prestel's mailbox frames lack, and this has always infuriated me.

Something else that infuriates me is that whilst on-line to Prestel, a user never knows when mail has entered his mailbox unless he checks it. On Epnitex, the status line at the bottom of the screen will inform you whenever mail arrives.

Epnilink is the name of the service primarily provided for business users. This will allow editing of frames by IPs with instant updating. This has great potential for companies wishing to transfer information to various offices around the country and operate a real-time display. It can also be used for transferring large amounts of data for updating a user's own computer database.

Micro enthusiasts will be glad, and hardly surprised I feel, to know that Epnitex will feature a large area devoted to computing. Timefame is well aware that micro owners have massively swelled the number of Prestel subscribers thanks to Micronet. Information on what exactly Epnitex will offer to micro enthusiasts was not forthcoming from Timefame, except that they will offer everything that Micronet is doing and much more.

The popular Microboard from Timefame's Prestel days will be a definite feature, with many enhancements. These will naturally mean a far larger area will be available than when operated on Prestel and with greater speed too.

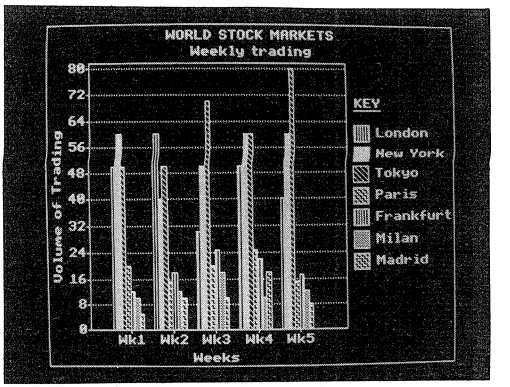
What equipment Timefame is using to implement Epnitex is another secret that they are keeping safe. Roy Norman would only say that a large investment has been made in hardware and software. Initially, the system will operate at two baud rates, 1200/75 as does Prestel, and at 1200/1200, the latter allowing both IP's and subscribers to transmit information back at a realistic speed. By February 1987, Timefame hope that a 2400/2400 baud service will be available too. Let's hope that the price of the hardware to provide access at 2400/2400 soon becomes realistic enough for domestic users to take advantage of such a service!

Two important questions to be answered are, how national access to the system at local call rates will be arranged, and what will be the cost of subscribing to Epnitex? Next month I should have the answers to these questions and full details of what Epnitex has to offer.

Timefame is making bold claims that their service will be significantly better than Prestel. To better Prestel in terms of facilities is not hard, as that system was created several years back and now shows its age. Timefame will have to convince both potential IPs and subscribers, that their extra facilities make Epnitex a worthwhile proposition. We shall see.

BUSINESS GRAPHICS (Part 3)

Alan Dickinson concludes the development of the BEEBUG Business Graphics system by adding two further display routines, and showing how data can be imported from other databases.



In the first of these three articles we presented a program to draw high quality business graphics on the BBC micro. In part two we added an easy-to-use screen-based editor. This final article of the series provides some refinements to the graphic capabilities (3D graphs and pie charts), and discusses ways in which you could develop the program further.

The additional code given this month should be appended to the program that you have already built up with parts one and two. To do this, follow the instructions given last month for appending part two to part one, or refer to the section on merging programs in your User Guide. If you have kept to all the original line numbers, then you should find no problems.

The additional graphics features provide 3D 'pillar' charts and pie charts (see illustrations). Both are outline charts only, i.e. patterning information is not used. The pie chart will show how data is shared between variables for the first 'ON' column. These features are already referred to within the screen-based editor, and you only need to delete line 1975 to activate these two remaining graphics features once the new code has been added (lines 3050 to 4090).

Lines 1950 to 1960 provide a routine to save a screen dump to disc. You could, for example, then enhance this image with a paintbox program, and perhaps create a presentation sequence of slide images. If you are using shadow RAM or similar, you will need to customise this routine to suit your own hardware. If you have a ROM capable of doing a screen to printer dump then you may prefer to replace the contents of this routine just with a call to PROCdis followed by the star command needed for your screen dump.

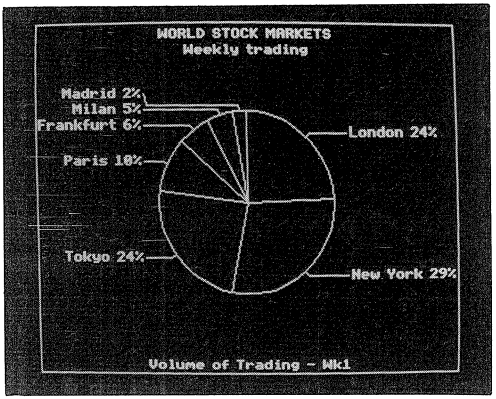
Further, lines 1800 to 1940 contain a routine to import a data file prepared using the BEEBUG spreadsheet program in Vol.3 Nos.9 & 10 (it only accepts data from the first 13 columns and 7 rows of the spreadsheet). You could, of course, write a similar routine to import data from any other file which contains information suitable for display in chart or graph form. It is the array N(12,6) - 13 by 7 - that holds the data. Rewriting the import routine to read any other data into this array will then allow it to be displayed with the Business Graphics program.

The complete program can just squeeze into a standard model B with DFS, hence the cryptic nature of variable names and the lack of comments. Users of an &E00 DFS, such as the Kenda DMFS, or a Master or Compact, will find that they have enough RAM left over to add new features such as customised import routines. Shadow RAM could also be used to enhance the program considerably. In particular, the code could be modified to use mode 1 (or 129) and produce full colour graphics.

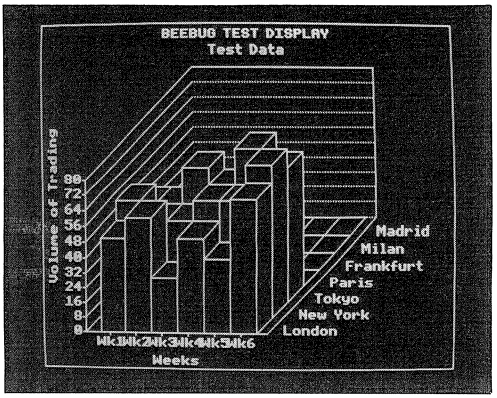
The graphics extension ROM (or the new graphics routines in the Master and Compact) could be put to good use by very much simplifying the patterning routines, and would allow pie charts and 3D charts to be patterned as well. Owners of a Master or Compact should be able to produce a very polished version indeed.

Note should be taken that this program has used the page of memory starting at &900. Cassette system users may prefer to change this to &D00. If memory permits, it would be better to DIMension a block of memory and use peeks and pokes in a legal and system independent manner.

```
 1800 DEFPROCimp:f$=FNzc("FILENAME",f$,8
):IFf$=""ENDPROC:ELSEA%=OPENUP(f$)
 1810 LOCALC%,P%,Q%,V%,X%,Y%,Z%:INPUT#A%
,X%,Y%
 1820 X%=X%-1:IFX%>12C%=12:ELSEC%=X%
 1830 Y%=Y%-1:IFY%>6V%=6:ELSEV%=Y%
 1840 FORZ%=0TO200:INPUT#A%,Z$:NEXT
 1850 FORP%=0TOV%:V%(P%)=1
 1860 FORQ%=0TOC%:C%(Q%)=1
 1870 INPUT#A%,N(Q%,P%)
 1880 NEXT:IFC%<X%FORQ%=C%+1TOX%:INPUT#A
%,Z%:NEXT
```




```
 1890 NEXT:IFV%<Y%FORP%=V%+1TOY%:INPUT#A
%,Z%:NEXT
 1900 FORQ%=0TOV%:INPUT#A%,V$(Q%):NEXT
 1910 IFV%<Y%FORQ%=V%+1TOY%:INPUT#A%,Z$:
NEXT
 1920 FORP%=0TOC%:INPUT#A%,C$(P%),Z%:NEX
T
 1930 IFC%<X%FORP%=C%+1TOX%:INPUT#A%,Z$,
Z%:NEXT
 1940 CLOSE#F%:ENDPROC
 1950 DEFPROCdump:D$=FNzc("FILENAME",D$,
8):IFD$=""ENDPROC
 1960 LOCALX%,Y%:PROCdis:$&900="SAVE "+D
$+" 5800 7FFF":X%=0:Y%=9:CALL&FFF7:ENDPR
OC
 2620 DEFPROCcube:GCOL0,1:s%=48*V%
 2630 MOVEP%,Q%:DRAWP%,S%:DRAWP%+s%,S%+s
%:DRAWR%+s%,S%+s%:DRAWR%+s%,Q%+s%:DRAWR%
,Q%:DRAWP%,Q%:ENDPROC
 3050 DEFPROCthreeD(n%)
 3060 x1%=P%+X*n%+s%:y1%=s%-48
 3070 FORK%=6TO0STEP-1
 3080 IFV%(K%)>0 PROCpillar(x1%,FNypos(0
)+y1%,x1%+X,FNypos(N(J%,K%))+y1%):x1%=x1
%-48:y1%=y1%-48
 3090 NEXT:ENDPROC
 3280 DEFPROCpie:GCOL0,1:LOCALA%,B%,X%,Y
%:A%=9999:B%=A%:x%=(P%+R%)/2:y%=(Q%+S%)/
2:r%=(S%-Q%)/2-128:MOVEx%,y%+r%
 3290 FORj=0TO2*PI STEPPI/31:x=x%+r%*SIN
(j):y=y%+r%*COS(j):DRAWx,y:NEXT:DRAWx%,y
%+r%:DRAWx%,y%
 3300 c=0:FORJ%=0TO6
 3310 IFV%(J%)>0PROCslice(J%)
 3320 NEXT:ENDPROC
 3330 DEFPROCslice(J%):LOCALI%,K%,L
 3340 pa=c*2*PI/csum:c=c+N(D%,J%)
 3350 a=c*2*PI/csum:L=(pa+a)/2
 3360 MOVEx%,y%:DRAWx%+r%*SIN(a),y%+r%*C
OS(a)
 3370 MOVEx%+r%*SIN(L),y%+r%*COS(L)
 3380 X%=x%+(r%+16)*SIN(L):Y%=y%+(r%+16)
*COS(L):DRAWX%,Y%:IFL>PI K%=-1 ELSEK%=1
 3390 X%=x%+(r%+32)*K%:DRAWX%,Y%:Y%=Y%+4
*K%:I%=4*K%:REPEAT:Y%=Y%-4*K%:I%=I%-4*K%
 3400 UNTIL(A%<>X%)OR((B%-Y%)*K%>40)
 3410 PLOT1,K%*8,I%:A%=X%:B%=Y%
 3420 pc%=N(D%,J%)/csum*100+0.5
 3430 p$=V$(J%)+" "+STR$(pc%)+"%":PLOT0,
8*K%,16:IFK%=1PROCtxt(p$,145,0,0)ELSEPRO
Ctxt(p$,161,0,0)
 3440 ENDPROC
 4050 DEFPROCpillar(A%,B%,P%,Q%)
 4060 GCOL0,0:MOVEA%,B%:MOVEP%,B%
 4070 PLOT85,A%,Q%:PLOT85,P%+48,B%+48:PL
OT85,A%+48,Q%+48:PLOT85,P%+48,Q%+48
 4080 GCOL0,1:MOVEA%,Q%:DRAWP%,Q%:DRAWP%
,B%:DRAWA%,B%:DRAWA%,Q%:DRAWA%+48,Q%+48:
DRAWP%+48,Q%+48:DRAWP%+48,B%+48:DRAWP%,B
%:MOVEP%,Q%:PLOT1,48,48
 4090 ENDPROC
```



Sideways RAM Buffers

If you constructed last month's sideways RAM module, or already have sideways RAM fitted to your machine, C. C. Radcliffe's multi-buffer utility will prove invaluable.

Sideways RAM is a popular add-on for many BBC micro users. Indeed, our constructional article last month (Vol.5 No.7) provides a very economic way for readers to add sideways RAM to their machine. Even so, many users may still wonder if sideways RAM can provide any real benefits.

One obvious, practical use is to extend one or more of the existing buffers (a buffer is an area of memory where the machine queues bytes to be sent to an external or internal I/O device, for example the printer, or the sound generator). Using sideways RAM in this way will greatly improve the performance of your machine, particularly when outputting to a printer.

The program listed in this article will allow the use of sideways RAM as a buffer for the keyboard, RS423 input or output, printer, sound channels 0 through 3, or speech. The maximum buffer size is over 15K, with 16K of RAM fitted, the rest of the sideways RAM being occupied by the program itself.

The program should work on any ROM board with RAM fitted, and has been tested on an ATPL Sidewise board, and with the BEEBUG sideways RAM module. It can be configured to suit either 8K or 16K of RAM, though the 8K should start at &8000.

USING THE PROGRAM

Type the program in and save it. Next, run the program, and if all is well there will be a pause while the machine code is assembled. This is followed by some requests for configuration data.

First you are asked to select the buffer size. Three options are offered:

Maximum with 16K SWRAM
Maximum with 8K SWRAM
Enter limits

Select your option with the cursor keys and then press Return. The first two options define the buffer area automatically as the end of the machine code up to &BFFF or &9FFF respectively. The third option allows the buffer limits to be specified by the user, but if the lower limit is set below the bottom address of the assembled program, the code will be corrupted.

You will then be asked to select the buffer you want to extend. All 9 buffers supported by the OS can be extended, but only one at a time.

The program will then save the code with the correct configuration data and an EXEC file. The EXEC file is intended for use with a !BOOT file or it may be called separately. When *EXECed, it will load the code, set the ROM type byte in the ROM information table, and switch on the buffer. Setting the ROM type means that the OS will recognise the presence of the code, stored in part of the sideways RAM.

The names of the files used depend on the buffer type selected as follows:

Buffer	M/C file	EXEC file
Keyboard	M.KBXBUF	$.KBXBUF
RS423 input	M.RIXBUF	$.RIXBUF
RS423 output	M.ROXBUF	$.ROXBUF
Printer	M.PRXBUF	$.PRXBUF
Sound ch.0	M.S0XBUF	$.S0XBUF
Sound ch.1	M.S1XBUF	$.S1XBUF
Sound ch.2	M.S2XBUF	$.S2XBUF
Sound ch.3	M.S3XBUF	$.S3XBUF
Speech	M.SPXBUF	$.SPXBUF

The code can also be loaded by *LOADing the appropriate file. If <Break> is then pressed, the OS will initialise the buffer code. However it is set up, an extended buffer can then be switched on and off with the commands *XBUF ON, and *XBUF OFF (abbreviations follow normal *command rules, and lower case is acceptable). Note, however, that the ON command re-initialises all the pointers, so any previously stored bytes will be

lost. If you wish to check in a program whether the buffer is empty, an OSBYTE call with A=&80 and X=255-(buffer number) will return the character or space count for input and output buffers respectively, in X and Y (lo-byte and hi-byte). Alternatively, an OSBYTE call with A=&98 with X=buffer number will return C=1 if the buffer is empty.

Those interested may wish to modify the program to extend several buffers at once (e.g. all 4 sound channels). This will require a table of pointers and more extensive buffer number checking at the start of each of the handling routines.

We appreciate that this is a particularly long program, but it is a most useful utility for all sideways RAM users. The program is included on this month's magazine cassette/disc.

```
  10 REM PROGRAM EXTENDED SWRAM BUFFER
  20 REM Version B1.1
  30 REM Author  C.C.Radcliffe
  40 REM BEEBUG  Jan/Feb 1987
  50 REM Program subject to copyright
  60 :
 100 REM.*** Assemble M/Code ***
 110 PROCassemble
 120 REM *** Configuration ***
 130 MODE7
 140 PROCinit
 150 PROCtitle
 160 PROCsize
 170 PROCbuffer
 180 PROCsave
 190 *FX4,0
 200 END
 210 :
1000 DEF PROCassemble
1010 LOCAL I%
1020 oswrch=&FFEE
1030 osnewl=&FFE7:osbyte=&FFF4
1040 insv=&22A:remv=&22C:cnpv=&22E:vect
=&0200:zp=&F6
1050 FOR I%=0 TO 3 STEP 3
1060 P%=&8000
1070 [OPT I%
1080 .ram:BRK:BRK:BRK
1090 JMP serviceentry
1100 OPT FNequb(&82)
1110 OPT FNequb(&13)
1120 OPT FNequb(&01)
1130 .title:OPT FNequs("XBUF")
1140 BRK:OPT FNequs(" 1.00")
1150 BRK:OPT FNequs("(C) 1986 BEEBUG"):
BRK
1160 .serviceentry:PHP:CMP #4
1170 BEQ unreccom:CMP #9:BEQ help
1180 PLP:RTS:.help:PHA:TXA:PHA:TYA
1190 PHA:JSR osnewl:LDA #&80:STA flags
1200 LDX #&FF:JMP unrecogloop:.helpmsg
1210 LDX #0:.titleloop:LDA title,X
1220 JSR oswrch:INX:CPX #10
1230 BNE titleloop:JSR osnewl
1240 JSR osnewl:LDX #0:.helploop
1250 LDA helptext,X:CMP #0
1260 BEQ linefeed:JSR oswrch:INX
1270 JMP helploop:.linefeed:JSR osnewl
1280 JSR osnewl:BIT flags:BVS out
1290 JMP exit:.out:LDA #0:STA flags
1300 PLA:TAY:PLA:TAX:PLA:PLP:RTS
1310 .unreccom:PHA:TXA:PHA:TYA:PHA
1320 LDX #&FF:.unrecogloop:INX
1330 LDA (&F2),Y:CMP #&41
1340 BMI notaletter:AND #&DF
1350 .notaletter:CMP title,X
1360 BNE abbrev:.paramloop:INY:CPX #3
1370 BNE unrecogloop:LDA flags
1380 BMI helpmsg:LDA (&F2),Y:CMP #&41
1390 BMI notaletter1:AND #&DF
1400 .notaletter1:CMP #13:BEQ restore
1410 CMP #&20:BEQ paramloop
1420 CMP #ASC("O"):BEQ paramloop
1430 CMP #ASC("F"):BEQ restore
1440 CMP #ASC("N"):BNE error
1450 JMP init:.abbrev:CMP #&0D
1460 BEQ helponly:CMP #&2E
1470 BNE notthisrom:CPX #1
1480 BMI notthisrom:LDX #3
1490 JMP paramloop:.notthisrom
1500 JMP out:.helponly:LDA #&C0
1510 STA flags:JMP helpmsg:.restore
1520 LDA flags:AND #1:BEQ exit
1530 LDX oldinsv:LDY oldinsv+1
1540 STX insv:STY insv+1:LDX oldremv
1550 LDY oldremv+1:STX remv:STY remv+1
1560 LDX oldcnpv:LDY oldcnpv+1
1570 STX cnpv:STY cnpv+1:.exit
1580 PLA:PLA:PLA:PLA:LDA flags
1590 AND #1:STA flags:LDA #0:RTS
1600 .error:LDX #0:.perrloop
1610 LDA parmerrmsg,X:JSR oswrch
1620 INX:CPX #17:BNE perrloop
1630 JSR osnewl:JMP helpmsg
1640 .parmerrmsg:OPT FNequs("Illegal Pa
rameter")
1650 .helptext:OPT FNequs("    Syntax *
XBUF ON/OFF"):OPT FNequb(&00)
1660 .init:LDA flags:AND #1
1670 BNE saveramno:LDX insv:LDY insv+1
1680 STX oldinsv:STY oldinsv+1
1690 LDX remv:LDY remv+1:STX oldremv
1700 STY oldremv+1:LDX cnpv:LDY cnpv+1
1710 STX oldcnpv:STY oldcnpv+1
1720 LDA flags:ORA #1:STA flags
1730 .saveramno:LDA &F4:STA swramno
1740 LDX#0:.initloop1:LDA mainvoff,X
```

```
1750 TAY:LDA extentoff,X:STA vect,Y
1760 INX:CPX #3:BNE initloop1
1770 .initloop2:LDA mainvoff,X
1780 TAY:LDA #&FF:STA vect,Y:INX
1790 CPX#6:BNE initloop2:LDA #&A8
1800 LDX #0:LDY #&FF:JSR osbyte
1810 STX extvecspace:STY extvecspace+1
1820 JSR zpsave:LDY #&FF:LDX #0:CLC
1830 LDA extvecspace:ADC extentoff
1840 STA zp:LDA extvecspace+1:ADC #0
1850 STA zp+1:.initloop3:INY
1860 LDA entrytab,X:STA (zp),Y:INX
1870 INY:LDA entrytab,X:STA (zp),Y
1880 INX:INY:LDA swramno:STA (zp),Y
1890 CPX #6:BNE initloop3:.bufinit
1900 LDA buftop:SEC:SBC bufstart
1910 STA space:LDA buftop+1
1920 SBC bufstart+1:STA space+1:CLC
1930 LDA space:ADC #1:STA space
1940 STA size:LDA space+1:ADC #0
1950 STA space+1:STA size+1
1960 LDA bufstart:STA remvptr
1970 STA insvptr:LDA bufstart+1
1980 STA remvptr+1:STA insvptr+1
1990 LDA flags:AND #2:BNE purge
2000 JMP exit:.purge:RTS:.zpsave:PHA
2010 LDA zp:STA tempzp:LDA zp+1
2020 STA tempzp+1:PLA:RTS
2030 .zprestore:PHA:LDA tempzp:STA zp
2040 LDA tempzp+1:STA zp+1:PLA:RTS
2050 .insert:SEI:CPX bufno:BEQ pbufins
2060 JMP (oldinsv):CLI:.pbufins:PHA
2070 SEC:LDA space:BNE notfull
2080 LDA space+1:BNE notfull1:PLA:CLI
2090 RTS:.notfull1:LDA space:.notfull
2100 SBC #1:STA space:LDA space+1
2110 SBC #0:STA space+1:JSR zpsave
2120 LDA insvptr:STA zp:LDA insvptr+1
2130 STA zp+1:LDY #0:PLA:STA (zp),Y
2140 JSR zprestore:PHA:LDA insvptr
2150 CMP buftop:BNE startinc
2160 LDA insvptr+1:CMP buftop+1
2170 BNE startincl:LDA bufstart
2180 STA insvptr:LDA bufstart+1
2190 STA insvptr+1:JMP insertexit
2200 .startincl:LDA insvptr
2210 .startinc:CLC:ADC #1:STA insvptr
2220 LDA insvptr+1:ADC #0:STA insvptr+1
2230 .insertexit:PLA:CLC:CLI:RTS
2240 .remove:SEI:PHP:CPX bufno
2250 BEQ pbufremv:PLP:CLI
2260 JMP (oldremv)
2270 .pbufremv:LDA remvptr:CMP insvptr
2280 BNE qexam:LDA remvptr+1
2290 CMP insvptr+1:BNE qexam:LDA space
2300 BNE empty:LDA space+1:BNE empty
2310 .qexam:PLP:BVC remvchr:JSR zpsave
2320 JSR setremvptr:LDA (zp),Y
2330 JSR zprestore:CLI:RTS:.empty:PLP
2340 SEC:CLI:RTS:.remvchr:CLC:LDA space
2350 ADC #1:STA space:LDA space+1
2360 ADC #0:STA space+1:JSR zpsave
2370 JSR setremvptr:LDA (zp),Y:TAY
2380 JSR zprestore:LDA remvptr
2390 CMP buftop:BNE ptrinc
2400 LDA remvptr+1:CMP buftop+1
2410 BNE ptrincl:LDA bufstart
2420 STA remvptr:LDA bufstart+1
2430 STA remvptr+1:JMP remvexit
2440 .ptrincl:LDA remvptr:.ptrinc
2450 CLC:ADC #1:STA remvptr
2460 LDA remvptr+1:ADC #0:STA remvptr+1
2470 .remvexit:CLC:TYA:CLI:RTS
2480 .setremvptr
2490 LDA remvptr:STA zp:LDA remvptr+1
2500 STA zp+1:LDY #0:RTS:.countp:SEI
2510 PHP:PHP:CPX bufno:BEQ pbufcnpv
2520 PLP:PLP:CLI:JMP (oldcnpv)
2530 .pbufcnpv:PLP:BVC count
2540 LDA flags:ORA #2:STA flags
2550 JSR bufinit:LDA flags:AND #&FD
2560 STA flags:PLP:CLI:RTS:.count
2570 BCC entries:LDX space:LDY space+1
2580 PLP:CLI:RTS:.entries:SEC:LDA size
2590 SBC space:TAX:LDA size+1
2600 SBC space+1:TAY:PLP:CLI:RTS
2610 .flags:OPT FNequb(&00)
2620 .bufno:OPT FNequb(&00)
2630 .oldinsv:OPT FNequw(&0000)
2640 .oldremv:OPT FNequw(&0000)
2650 .oldcnpv:OPT FNequw(&0000)
2660 .swramno:OPT FNequb(&00)
2670 .extentoff:OPT FNequw(&423F)
2680 OPT FNequb(&45)
2690 .mainvoff:OPT FNequd(&2B2E2C2A)
2700 OPT FNequw(&2D2F)
2710 .extvecspace:OPT FNequw(&0000)
2720 .entrytab:OPT FNequd(&00000000)
2730 OPT FNequw(&0000)
2740 .tempzp:OPT FNequw(&0000)
2750 .space:OPT FNequw(&0000)
2760 .remvptr:OPT FNequw(&0000)
2770 .insvptr:OPT FNequw(&0000)
2780 .size:OPT FNequw(&0000)
2790 .buftop:OPT FNequw(&0000)
2800 .bufstart:OPT FNequw(&0000)
2810 .endcode
2820 ]
2830 NEXT
2840 ?entrytab=insert MOD 256
2850 entrytab?1=insert DIV 256
2860 entrytab?2=remove MOD 256
2870 entrytab?3=remove DIV 256
2880 entrytab?4=countp MOD 256
2890 entrytab?5=countp DIV 256
2900 ?buftop=&FF:buftop?1=&BF:?bufstart
=endcode MOD 256:bufstart?1=endcode DIV
256
2910 ?bufno=3
2920 ENDPROC
```

```
2930 :
2940 DEF PROCinit
2950 LOCAL N%
2960 DIM text$(11)
2970 FOR N%=0 TO 11
2980 READ text$(N%)
2990 NEXT
3000 *FX4,1
3010 key$=CHR$(&8B)+CHR$(&8A)+CHR$(&0D)
3020 hex$="0123456789ABCDEF"+CHR$(127)+
CHR$(&0D)
3030 ENDPROC
3040 :
3050 DEF PROCtitle
3060 PRINT CHR$133;CHR$157;CHR$130;TAB(
9)CHR$141;"BUFFER CONFIGURATION"
3070 PRINT CHR$133;CHR$157;CHR$130;TAB(
9)CHR$141;"BUFFER CONFIGURATION"
3080 PRINT CHR$133;CHR$157;CHR$130;TAB(
10);STRING$(20,"*")
3090 ENDPROC
3100 :
3110 DEF PROCsize
3120 LOCAL option%
3130 PRINTTAB(1,4)"Select Buffer Size :"
3140 option%=FNselect(0,3,6,1,134)
3150 ON option% GOSUB 3170,3170,3190
3160 ENDPROC
3170 PROCsetsize(16 DIV option%)
3180 RETURN
3190 PROCgetsize
3200 RETURN
3210 :
3220 DEF PROCbuffer
3230 LOCAL option%
3240 PRINTTAB(1,10)"Select Buffer Number :"
3250 option%=FNselect(3,9,12,4,131)
3260 ?bufno=option%-1:O%=option%
3270 ENDPROC
3280 :
3290 DEF PROCsave
3300 LOCAL N%,R%,P$
3310 RESTORE 4770
3320 R%=FNsocket
3330 FOR N%=1 TO O%
3340 READ P$
3350 NEXT
3360 PROCdownload(R%)
3370 PROCcodesave(P$)
3380 PROCexecfile(P$,FNromtabadd,FNromt
ype(R%))
3390 ENDPROC
3400 :
3410 DEF FNselect(textno%,lines%,startl
ine%,default%,colour%)
3420 LOCAL L%,N%,M%,C%,K%,K$
3430 M%=startline%
3440 FOR N%=textno% TO textno%+lines%-1
3450 PRINTTAB(5,M%);CHR$(colour%);text$
(N%);SPC(2);CHR$(156)
3460 M%=M%+1
3470 NEXT
3480 C%=default%-1:L%=C%
3490 REPEAT
3500 PRINTTAB(3,startline%+L%);CHR$32;C
HR$156;CHR$(colour%);
3510 PRINTTAB(3,startline%+C%);CHR$129;
CHR$157;CHR$(colour%);
3520 REM PRINTTAB(LEN(text$(textno%+C%)
)+8,startline%+C%)CHR$156
3530 REPEAT
3540 K%=FNkey
3550 UNTIL ((C%+K%)>=0 AND (C%+K%)<line
s%) OR K%=3
3560 IF NOT(K%=3) L%=C%:C%=C%+K%
3570 UNTIL K%=3
3580 =C%+1
3590 :
3600 DEF FNkey
3610 LOCAL K$,K%
3620 REPEAT
3630 K%=GET
3640 K$=CHR$(K%)
3650 K%=INSTR(key$,K$)
3660 UNTIL K%
3670 IF K%=1 =-1
3680 IF K%=2 =1
3690 =K%
3700 :
3710 DEF PROCsetsize(S%)
3720 LOCAL T%
3730 ?bufstart=endcode MOD 256:bufstart
?1=endcode DIV 256
3740 T%=&7FFF+S%*1024
3750 ?buftop=T% MOD 256:buftop?1=T% DIV
256
3760 ENDPROC
3770 :
3780 DEF PROCgetsize
3790 LOCAL N%,B%,T%
3800 FOR N%=4 TO 8
3810 PRINTTAB(0,N%)STRING$(40," ")
3820 NEXT
3830 PRINTTAB(2,4)CHR$(134);"Enter Buff
er Bottom Address in HEX,"
3840 PRINTTAB(2)CHR$(134);"(&";~endcode
;" is lowest available)";CHR$(133);"&";
3850 B%=FNHEX(32,5)
3860 PRINTTAB(2,6)CHR$(134);"Enter Buff
er Top Address,"
3870 PRINTTAB(2)CHR$(134);"(&BFFF is hi
ghest available)";CHR$(133);"&";
3880 T%=FNHEX(33,7)
3890 ?bufstart=B% MOD 256:bufstart?1=B%
DIV 256
3900 ?buftop=T% MOD 256:buftop?1=T% DIV
256
3910 ENDPROC
3920 :
3930 DEF FNHEX(X%,Y%)
```

→28

A Penfriend for Wordwise

The Wordwise Plus programming language gave rise to a number of supporting Wordwise Plus utilities, including our own Wordease. Can a relative newcomer to this field have anything new to offer? Geoff Bains has been finding out.

Product: Penfriend
Supplier: Word Processing
P.O. Box 67, Wolverhampton,
West Midlands.
Price: £14.95 inc. VAT

The aptly-named Pen-Friend is a Wordwise Plus utility ROM. This in itself is nothing unusual; there are many such utilities around. However, what distinguishes Pen-Friend from the rest is that the utilities provided are accessed from within edit mode using a menu screen that, rather confusingly, overlays part of the text area.

The Pen-Friend ROM is partly machine code and partly Wordwise Plus programs - stored as ROM filing system files. The entire software makes extensive use of the segments provided in Wordwise Plus.

The ROM is initialised with *PEN. This loads the main menu routine into segment number one, selects edit mode and runs the menu. The menu takes up about a quarter of the screen, overlapping the text screen. After loading, the menu can be run by pressing Shift-f1.

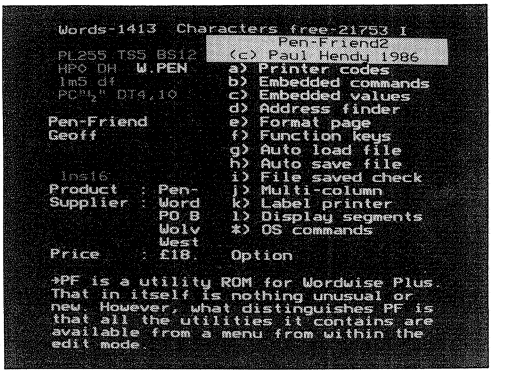
All the other routines are accessed via this menu. There are 13 options available from the menu. Some of these are definitely useful; a few are simply baffling as to why they are included.

The first option is to embed printer control codes in a document. This produces a secondary menu allowing underline, double strike, bold, Elite, and enlarged effects to be switched on or off. In itself that is pretty limp. However, three further options allow either a default Brother (daisy wheel), Epson (dot matrix), or user defined 'printer drivers' to be loaded from the ROM or from disc.

All these effects make use of the preset printer sequence facility of Wordwise Plus. The 'drivers' merely define the effect of these codes at the beginning of a document.

It has to be said an equally efficient effect can be produced by using a standard document header (which many users do anyway) including suitable code definitions and inserting the code sequences 'by hand'. However, Pen-Friend does make it all marginally less arduous.

The second option on the menu is to insert embedded commands at the cursor position. Again this is stretching the 'ease-of-use' aspect of Pen-Friend a bit far. All this means is that you can forget codes needed for 14 embedded functions. It could be argued that these are either commonly used and so quickly memorised anyway, or more rarely used and therefore less useful for inclusion in Pen-Friend. I would prefer to see just a pull down help screen of all the embedded commands.



The next option is the 'Status' display. This is genuinely very useful. The routine produces a display of the format of the finished document at the cursor position - the indentation, left and right margins, page length, top and bottom space, header and footer positions,

pad character, pound sign output character, page number, line number, cursor position, and the status of the justification, paged display and message display flags.

All this is done without recourse to the preview mode. When near the end of a long document, this routine swiftly goes a long way to getting around the what-you-see-is-far-from-what-you-get drawback of Wordwise Plus.

The next routine provided by Pen-Friend's main menu is the 'address finder'. This is a simple card index type of a database with automatic insertion of entries found from a search into a document. The addresses must be entered into five alphabetically sorted Wordwise files in a specific format. Rather strangely, blank lines in the addresses must be filled up with @ symbols.

When the address finder option is selected and a search string (including a wild card if required) entered, the files are searched and the matching addresses entered at the cursor position. All this works well enough, but it seems unnecessary for most users.

The 'Format page' option also seems too much trouble for so little gain. Like the embedded command routine, this one displays active formatting commands in a document at the cursor position, and inserts a new set at the top of the document according to entered parameters.

The biggest problem is that if this routine is called twice, to again alter the format, the new set of commands is then inserted at the top of the document but their effect is immediately cancelled by the first set which still remain.

The next option is the function key option. This uses a file of several sets of function key definitions (created with Wordwise), each set labelled with a comment and a number. The routine displays all the comments and allows a set to be selected and loaded.

The next two options allow easier loading and saving of documents. The auto-load routine displays a menu of the available files to be selected with a single key press. The auto-save routine will save the latest version of your document with a new name created by incrementing the last letter through A-Z and 1-9.

However, the next option is another gem. Option J will reformat Wordwise Plus text into a multi-column document. Up to five columns of text across a page are possible. This is not up to the standard of the InterWord multi-column facility but it is well done nonetheless. Unlike the dual column routine that accompanies Wordwise Plus itself, this routine does not simply divide the text in the middle but makes each column on a page read on from the previous one, just like a newspaper (or, indeed, BEEBUG).

There are two further options worth noting, both providing useful facilities for Wordwise Plus users. Option K takes the address files used for the address finder and will print out address labels from them on fanfold label stationary. Option L displays a resume of the contents of the ten segment areas - like the numerous ROM index utilities around. The final option is to access * commands. Now this really is silly. It takes two key presses before a star command is actually typed in (Shift-f1 for the menu and * to select this option) and two after it has done its stuff to return to the text (any key then Escape). The normal way of issuing * commands requires Escape to get to command mode then * and afterwards any key followed by Escape to get back to the edit mode. That's right: two plus two key presses. Just the same. So why the new way?

There is no doubting that Pen-Friend is a well written piece of software and at £19 it is reasonably priced as Beeb ROMs go. I do, however, question its value to many Wordwise Plus users. Anyone, with any experience of Wordwise Plus, can perform most of the functions Pen-Friend provides just as easily (and usually as quickly) without it. More amazingly, for a product for word processing, from an outfit of the same name, the manual is littered with mistakes, mis-spellings, and poor editing.

There are some good points. The multi-column and status routines are excellent. The auto-save and function key routines are reasonable. However, this is not enough to justify the Pen-Friend ROM.

VIRTUAL ARRAYS

(Part 1)

Large arrays can consume precious memory at an alarming rate. Jacob Abboud shows how really large arrays can be used in a program by storing them on disc, or in sideways RAM.

The limited memory available for the programmer on a BBC micro is a major disadvantage when developing serious software. This is most evident when developing CAD programs or programs which require large array storage. The problem can be overcome by using a second processor (an expensive solution!), or a shadow RAM board which increases the available program memory irrespective of the mode used. Alternatively, procedure overlay techniques can be used to reduce the overall program size and allow more variables and arrays to be stored.

During the development of a program, I was faced with the problem of trying to run a 13K program which requires 38K of array storage and, to make things worse, uses mode 0 for graphical output. The solution I came up with is simple, and can be implemented using what is called the 'Virtual Memory' storage technique. This method is widely used on mainframes and minis to make a computer's memory appear larger than it really is. This illusion is achieved on mainframes by dividing the contents of memory into 'pages' and storing these pages on disc (hard disc). When a program refers to a page which is not in memory, the program is temporarily suspended, and the missing page is read in as quickly as possible. When a program modifies the data on a page, the page is copied out to the backing store. The operating system attempts to meet the needs of a program by judiciously moving the pages in and out of main memory.

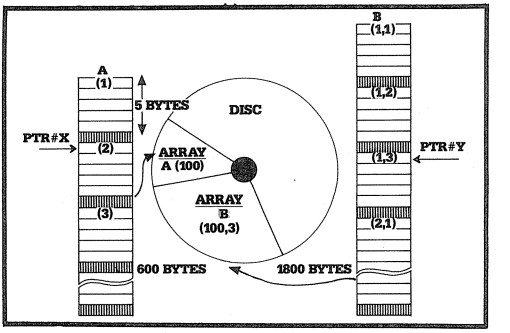
Although a full implementation of the principle is impractical on a BBC micro, since hardware and software (operating system) modifications are required, the technique can nevertheless be used for storing large arrays. This method is most suited for RAM discs, which are becoming more popular amongst many users because of fast access times, though it will also work with normal disc drives, albeit with a time penalty.

Suppose a program uses three floating point arrays X, Y, and Z, which are dimensioned X(500), Y(500), Z(500,3). The memory required for these arrays is 14.7K, since a real number on the BBC micro occupies 5 bytes. This will obviously leave little room for the program and screen RAM, and one is left with the dreaded 'No room' error message. The problem can be overcome by using the proposed method of virtual storage.

The virtual memory technique can be implemented in one of three ways:

1. Separate random access files.
2. Stacked random access file.
3. Paged array/files.

Each of the above methods has its merits and limitations. These will be discussed in due course. This month we will look at the first method.



SEPARATE RANDOM ACCESS FILES

With this method each array is stored in a separate disc file. The basic technique requires a virtual memory driver to be installed before running the main program. Access to the elements of each array is achieved by the inclusion of three procedures within the main program. To install the driver, type in Listing 1 and save a copy first.

When it is run, the program will enquire whether a RAM disc is in use, and if so issues the appropriate command (*RAMDFS <drive>) to initialise it (incidentally, I use the Solidisk 256

board which runs at 4MHz, and the Solidisk DFS2.2; if you use a different RAM disc replace the command at line 270 by the appropriate command). The program then asks for the number of arrays to be used (less than or equal to five). Each array is identified by a reference number (e.g. 1,2,3 ...), its dimension(s), and type (e.g. integer or real).

After the information for each array has been entered, it is displayed on the screen for verification, and the process is repeated for each selected array. On completion, the program assigns a letter (A,B,C,..) as a file name for each array before initialising it on the selected disc. If you are not using a RAM disc, place a formatted disc (DFS or ADFS) in the selected drive.

In order to demonstrate the method, let's install a driver for two arrays (integer or real) A(100), and B(100,3). Once the run is completed you can check that files A and B have been written to disc. You should also find that a third file !info has been created. This file contains information which is necessary to link the driver to the main program.

The next step is to type in Listing 2 and save a copy. This program is an example of the use of the procedure. It demonstrates how the procedures PROCopen, FNin, and PROCout should be incorporated in your programs, and how to access the elements of the arrays on disc. PROCopen, called at the beginning of the program, reads the information in !info into locations &70-&89 and opens the necessary files for input/output (OPENIN). PROCout handles the process of writing data to disc whilst FNin reads data from disc.

Accessing an element of an array, say A(10,30)=H, is transformed into PROCout(10,30,H,"A"). The first array specified is always given the name A, the second B, and so on. PROCout sets the pointer of the file named and then writes the data into that location in the file. Similarly, to read A(10,30) into H, use H=FNin(10,30,"A"), which again sets the pointer of the file referenced and reads the data. If one-dimensional arrays are used, the second dimension should be set to 1. Figure 1 is a schematic representation of files A and B. The demonstration program writes to each location of arrays A and B defined above, and then reads the stored values. A timer is incorporated to assess the time taken by each step. Table 1 shows the time taken by RAM, RAM disc, and normal disc.

HARDWARE	WRITE	READ
RAM	4	3 secs
RAM disc	7	4 secs
Disc	11	5 secs

Table 1. Time to read 400 bytes

Although string arrays are not included in the programs, they can easily be incorporated in the same way provided that each string in an array is of the same length. This method is very efficient for random access and is very easy to use. However, the maximum number of arrays (one or two dimensional) that can be used at any time depends on the filing system; 5 for the DFS but 10 for the ADFS. The maximums are set at line 290 in the driver program and at line 1030 in the demonstration program.

```
  10 REM Program VIRTUAL MEMORY DRIVER
  20 REM Version B2.4
  30 REM Author  Jacob Abboud
  40 REM BEEBUG  Jan/Feb 1987
  50 REM Program subject to copyright
  60 :
 100 DIM IN(5,5)
 110 @%=3:s1%=0:s2%=1:ram%=0
 120 FOR I%=1 TO 5:IN(I%,4)=1:NEXT
 130 MODE7
 140 ONERROR PROCerr:CLS
 150 FOR I%=2TO3
 160 PRINT TAB(1,I%);CHR$141;"VIRTUAL M
EMORY DRIVER"
 170 NEXT
 180 PRINT'"  Version 2.4"
 190 PRINT'"  RAMDFS (Y/N)     ?";
 200 A$=FNget("YN")
 210 IF A$="Y" ram%=1:C$="R":GOTO 240
 220 PRINT'"  (D)FS or (A)DFS ?";
 230 C$=FNget("AD")
 240 PRINT'"  Which DRIVE      ?";
 250 A$=FNget("0123"):D%=VAL(A$)
 260 IFC$="D" PROCoscli("DRIVE"+A$) ELS
E IF C$="A" PROCoscli("MOUNT"+A$)
 270 IF ram%=1 PROCoscli("RAMDFS"+A$)
 280 PRINT'"  HOW MANY ARRAYS ?";
 290 A$=FNget("123456789")
```

```
  300 VDU28,0,24,39,VPOS+1
  310 n%=VAL(A$)
  320 FOR I%=1 TO n%
  330 CLS:PRINT''SPC12;:VDU&87,&9D,&84:P
RINT;"ARRAY (";I%;")";"  ";CHR$&9C;" "'
  340 PRINT"  (S)ingle or (D)ouble ?";:A
$=FNget("SD")
  350 IF A$="S" IN(I%,1)=1 ELSE IN(I%,1)
=2
  360 PRINT'"  (I)integer or (R)eal ?";:
A$=FNget("IR")
  370 IF A$="I" IN(I%,2)=0:it$="INTEGER"
:PRINT
  380 IF A$="R" IN(I%,2)=1:it$="REAL":PR
INT
  390 PRINT"  Enter size of array  ";:IN
PUTs1%:IN(I%,3)=s1%
  400 IF IN(I%,1)=2 PRINT"  Second dimen
sion     ";:INPUTs2%:IN(I%,4)=s2%
  410 CLS
  420 PRINT'"  ARRAY";SPC(12);I%
  430 PRINT"  NAME";SPC(13);CHR$(64+I%)
  440 PRINT"  FIRST DIMENSION  ";s1%
  450 PRINT"  SECOND DIMENSION ";s2%
  460 PRINT"  ARRAY TYPE";SPC(7);"<";it$
;">"
  470 PRINT'"  Correct (Y/N) ?";
  480 A$=FNget("YN")
  490 IF A$="N" GOTO 330
  500 NEXT
  510 PROCinit
  520 PRINT'"  Virtual Memory Installed.
.."
  530 END
  540 :
 1000 DEF PROCinit
 1010 PRINT'"  Initialising ..."
 1020 LOCAL K%,J%,I%
 1030 Z=OPENOUT("!info")
 1040 BPUT#Z,n%
 1050 FOR I%=1 TO n%
 1060 dd%=IN(I%,1):it%=IN(I%,2)
 1070 s1%=IN(I%,3):s2%=IN(I%,4)
 1080 X=OPENOUT(CHR$(64+I%))
 1090 FOR K%=1 TO s1%
 1100 IF dd%=2 FOR J%=1 TO s2%
 1110 IF it%=0 PRINT#X,0
 1120 IF it%=1 PRINT#X,1E-20
 1130 IF dd%=2 NEXTJ%
 1140 IF K% MOD 50=0 PRINT TAB(15+5*I%,V
POS-1);K%
 1150 NEXT K%
 1160 CLOSE#X
 1170 BPUT#Z,64+I%:BPUT#Z,s2%:BPUT#Z,it%
 1180 NEXTI%
 1190 CLOSE#Z
 1200 ENDPROC
 1210 :
 1220 DEF PROCerr
 1230 IF ERR<>191 GOTO 1320
```

Driver Program

PROCEDURES AND FUNCTIONS

init	Initialises array files on disc and generate !info file.
err	Error handling routine. If Disc error 191 option for reformatting disc. Change *F80 <drive> to appropriate command.
get	Selective input function.
oscli	Passes command to line interpreter.

VARIABLES

IN()	Array specification.
s1%	First dimension.
s2%	Second dimension.
ram%	Flag for RAM disc.
D%	Drive number.
A$	Input string.
n%	Number of arrays.
it%	Array type.
dd%	single or two dimensional array marker.
B$	OS command used by oscli.

Demonstration Program

PROCEDURES AND FUNCTIONS

open	Reads !info, stores array. data, and open relevant files for input/output.
out	Writes data to array.
in	Reads data from array.
time	Timer, time elapsed in seconds.

VARIABLES

n%	Number of arrays.
f%,a%	Array reference number.
f$	Array name.
i%	First array dimension index.
j%	Second array dimension index.
b,b%	Number to be written by PROCout.
v,v%	Number read by FNin.
N%	Number of bytes to be written.
I%,J%	Counters.

```
 1240 PRINT'"  RAMDISC EXISTS!"
 1250 PRINT'"  Reformat (Y/N) ?";
 1260 A$=FNget("YN")
 1270 IF A$="N" CLS:END
 1280 *ENABLE
 1290 PROCoscli("F80 "+STR$(D%))
 1300 PRINT "  Formatting completed"
 1310 ENDPROC
 1320 PRINT'"  *** ERROR ";ERR;" ***"
 1330 REPORT
 1340 PRINT" at line ";ERL
 1350 END
```

```
1360 :
1370 DEF FNget(B$)
1380 *FX20 0
1390 REPEAT K=GET AND 223:IFK<26 K=K+32
1400 K$=CHR$K:UNTIL INSTR(B$,K$)
1410 PRINTK$;:=K$
1420 :
1430 DEF PROCoscli(B$)
1440 LOCAL X%,Y%
1450 $&700=B$
1460 X%=0:Y%=7
1470 CALL &FFF7
1480 ENDPROC
```

```
  10 REM Program VIRTUAL ARRAY DEMO
  20 REM Version B1.8
  30 REM Author  Jacob Abboud
  40 REM BEEBUG  Jan/Feb 1987
  50 REM Program subject to copyright
  60 :
 100 MODE7
 110 @%=4
 120 PROCopen
 130 TIME=0
 140 PRINT"Writing data to arrays"'
 150 FOR I%=1 TO 100
 160 PROCout(I%,1,I%,"A")
 170 PRINT TAB(1,8);I%
 180 NEXT
 190 FOR I%=1 TO 100
 200 FOR J%=1 TO 3
 210 PROCout(I%,J%,I%,"B")
 220 PRINT TAB(1,9);I%,J%
 230 NEXT
 240 NEXT
 250 N%=400
 260 PRINT';N%;" Numbers written to VIR
TUAL MEMORY"
 270 PROCtime:TIME=0
 280 :
 290 PRINT'"Reading data ..."'
 300 I%=0
 310 REPEAT
 320 I%=I%+1
 330 PRINT TAB(1,18);I%,FNin(I%,1,"A")
 340 UNTIL I%=100
 350 :
 360 I%=0:J%=0
 370 REPEAT
 380 I%=I%+1:J%=J%+1
 390 PRINT TAB(1,19);I%,J%,FNin(I%,J%,"
B"):IF J%=3 J%=1
 400 UNTIL I%=100
 410 :
 420 PROCtime
 430 CLOSE#0
 440 END
 450 :
1000 DEF PROCopen
1010 Z=OPENIN("!info")
1020 n%=BGET#Z
1030 IF n%>5 PRINT"Too many files":END
1040 FOR I%=1 TO n%
1050 f%=BGET#Z:f$=CHR$f%
1060 I%?&6F=BGET#Z
1070 I%?&7F=BGET#Z
1080 IF I%?&7F=0 I%?&7F=5 ELSE I%?&79=6
1090 PRINT"Opened file ";f$
1100 I%?&84=OPENIN(f$)
1110 NEXT
1120 PRINT'"> Ready ..."''
1130 ENDPROC
1140 :
1150 DEF PROCout(i%,j%,b,A$)
1160 a%=ASC(A$)-64
1170 of%=a%?&6F*a%?&7F*(i%-1)+a%?&7F*(j
%-1)
1180 PTR#a%?&84=of%
1190 IF a%?&7F=6 PRINT#a%?&84,b
1200 IF a%?&7F=5 b%=b:PRINT#a%?&84,b%
1210 ENDPROC
1220 :
1230 DEF FNin(i%,j%,A$)
1240 a%=ASC(A$)-64
1250 of%=a%?&6F*a%?&7F*(i%-1)+a%?&7F*(j
%-1)
1260 PTR#a%?&84=of%
1270 IF a%?&7F=5 INPUT#a%?&84,v%:=v%
1280 IF a%?&7F=6 INPUT#a%?&84,v:=v
1290 :
1300 DEF PROCtime
1310 time=(TIME DIV 100)MOD 60
1320 PRINT'"Time taken = ";time;" secon
ds"
1330 ENDPROC
```

POINTS ARISING POINTS ARISING POINTS ARISING POINTS

BEEBUGSOFT Forum (BEEBUG Vol.5 No.6)

Under the heading 'Magscan - Split Files' it is essential to include a space in front of the '2500' at line 1290. This should read:

```
1290 IF R%<>I% $S="L.VOL"+CHR$(I%+48)+" 2500":X%=&40:Y%=7:CALL&7FFF
```

```
 3940 LOCAL A$,H$
 3950 REPEAT
 3960 H$=FNhex
 3970 IF ASC(H$)=127 A$=LEFT$(A$,LEN(A$)
-1):PRINTTAB(X%,Y%);STRING$(4," ")
 3980 IF NOT(ASC(H$)=13 OR ASC(H$)=127)
AND LEN(A$)<4 A$=A$+H$
 3990 PRINTTAB(X%,Y%);A$
 4000 UNTIL ASC(H$)=13 AND LEN(A$)=4
 4010 =EVAL("&"+A$)
 4020 :
 4030 DEF FNhex
 4040 LOCAL K$
 4050 REPEAT
 4060 K$=GET$
 4070 UNTIL INSTR(hex$,K$)
 4080 =K$
 4090 :
 4100 DATA "Maximum with 16K SWRAM","Max
imum with 8K SWRAM","Enter limits","0-Ke
yboard","1-RS423 input","2-RS423 output"
,"3-Printer"
 4110 DATA "4-Sound channel 0","5-Sound
channel 1","6-Sound channel 2","7-Sound
channel 3","8-Speech"
 4120 END
 4130 :
 4140 DEF FNsocket
 4150 LOCAL N$,N%,X%,Y%,A%,C%
 4160 Y%=&10
 4170 REPEAT
 4180 Y%=Y%-1
 4190 N$=""
 4200 FOR N%=&9 TO &C
 4210 !&F6=N%+&8000
 4220 N$=N$+CHR$((USR(&FFB9) AND &FF))
 4230 NEXT
 4240 UNTIL N$="XBUF"
 4250 =Y%
 4260 :
 4270 DEF PROCdownload(S%)
 4280 LOCAL A%,B%,C%,N%,X%,Y%
 4290 PRINTTAB(5,22)"Downloading and sav
ing Code"
 4300 PRINTTAB(5)"Current Address :"
 4310 VDU 28,25,23,35,23
 4320 Lcode%=endcode-&8001
 4330 Y%=S%
 4340 FOR N%=0 TO Lcode%
 4350 !&F6=N%+&8000
 4360 B%=USR(&FFB9) AND &FF
 4370 N%?&6000=B%
 4380 IF (N% MOD &40)=0 PRINTTAB(0,0)"&"
;~(N%+&8000);
 4390 NEXT
 4400 VDU26,31,0,24
 4410 ENDPROC
 4420 :
 4430 DEF PROCcodesave(P$)
 4440 LOCAL X%,Y%
 4450 $&900=("SAVE M."+P$+"XBUF 6000 +"+
STR$~(Lcode%+1)+" 8000 8000"):X%=0:Y%=9:
CALL &FFF7
 4460 ENDPROC
 4470 :
 4480 DEF PROCexecfile(P$,A$,T$)
 4490 LOCAL X%,I%,N%,S$
 4500 X%=OPENOUT(P$+"XBUF")
 4510 FOR I%=0 TO 2
 4520 IF I%=0 S$="*LOAD M."+P$+"XBUF"
 4530 IF I%=1 S$="?&"+A$+"=&"+T$
 4540 IF I%=2 S$="*XBUF ON"
 4550 FOR N%=1 TO LEN(S$)
 4560 BPUT#X%,ASC(MID$(S$,N%,1))
 4570 NEXT
 4580 BPUT#X%,13
 4590 NEXT
 4600 CLOSE #X%
 4610 ENDPROC
 4620 :
 4630 DEF FNromtabadd
 4640 LOCAL P%,X%,Y%,A%
 4650 A%=&AA:X%=0:Y%=&FF
 4660 P%=USR(osbyte)
 4670 P%=(P% AND &FFFF00) DIV &100
 4680 P%=P%+R%
 4690 =STR$~(P%)
 4700 :
 4710 DEF FNromtype(R%)
 4720 LOCALX%,Y%,A%,C%
 4730 Y%=R%
 4740 !&F6=&8006
 4750 =STR$~((USR(&FFB9) AND &FF))
 4760 :
 4770 DATA KB,RI,RO,PR,S0,S1,S2,S3,SP
 4780 :
 4790 DEF FNequb(byte%)
 4800 ?P%=byte%
 4810 P%=P%+1
 4820 =I%
 4830 :
 4840 DEF FNequw(word%)
 4850 ?P%=word% MOD 256
 4860 P%?1=word% DIV 256
 4870 P%=P%+2
 4880 =I%
 4890 :
 4900 DEF FNequd(dword%)
 4910 ?P%=dword% MOD &100
 4920 P%?1=dword% DIV &100
 4930 P%?2=dword% DIV &10000
 4940 P%?3=dword% DIV &1000000
 4950 P%=P%+4
 4960 =I%
 4970 :
 4980 DEF FNequs(string$)
 4990 $P%=string$
 5000 P%=P%+LEN(string$)
 5010 =I%
```

←21

Assembly Language Programming on the BBC and Acorn Electron

R.B. Coats

This book, which can be used with both the BBC and Acorn Electron computer will appeal to those teaching assembly language programming as part of a computer studies course, and their students, who want to extend their knowledge beyond BASIC into assembly language.

1985 288 pages illustrated
£12 paper 0 7131 3550 6

Tracer

R.B. Coats

A special program to help in learning assembly language, developed by the author for use with his book. Available on disk (40 track) for Model B and Electron.

£8.50 + VAT

BBC Basic

R.B. Coats

Using simple examples throughout, the reader is taught to program in BASIC.

1983 256 pages illustrated
£7.30 paper 0 7131 3497 6

MICROTAB – an all-purpose statistical package

P.G. Higginbotham

This comprehensive integrated software package for statistics and data analysis is designed to run on the BBC (Model B) microcomputer, easy for beginners and experts.

1985 Floppy disk and 96 pages manual £33 + VAT
80 track 0 7131 3601 8 40/80 track 0 7131 3549 2
Additional manuals are available from the publisher
£5.50 0 7131 3560 3

Advanced Statistical Analysis on BBC Microcomputers

N.W. Please

Probable publication February 1987 208 pages approx
£12.50 approx (book and disk) 0 7131 3566 2

All prices include postage and package.
Orders with payment to:
Edward Arnold (Publishers) Ltd
Woodlands Park Avenue
Maidenhead
Berkshire

For credit card facilities 062 882 3104

COMMAND — the ultimate communications ROM — for the BBC Micro & Master

Command is the ideal companion for the Beebug Magic Modem, and other BBC compatible modems.

Command is a very special command driven communications ROM, with a powerful extended instruction set. Because it may be command driven, it is exceptionally easy to link instructions together in Basic to build your own customised communications software.

- **Viewdata Terminal**
  A full feature Prestel Terminal with pull-down help screen, real time clock and Epson compatible mode 7 screen dump.

- **Text Terminal**
  Send files to a friend for the cost of a phone call, and access hundreds of Bulletin Boards throughout the country. Features include: 40/80 column operation, Xon/Xoff protocol, and pull-down status panel showing how Command is configured.

- **Telephone Directory**
  This excellent facility allows you to save a name, number and modem configuration onto disc, for easy recall at a later date. No need to remember telephone numbers anymore.

- **Viewdata Editor**
  A full feature teletext editor, with a full range of editing commands, pull-down help window, and a pixel editor.

Over 50 powerful commands, including

★ ANSWER	★ BAND
★ CALL	★ CONNECT
★ DIRECT	★ DISCONNECT
★ DISPLAY	★ DOWNLOAD
★ ECHON	★ GRAB
★ LISTEN	★ PAUSE
★ PRON	★ PSCREEN
★ RETRY	★ RINGS
★ SAY	★ SDUMP
★ SEND	★ STANDARD
★ STAT	★ TEXT
★ UPLOAD	★ VEDIT
★ VIEWDATA	★ XON

PRICE	MEMBERS PRICE
£39.00	£29.25

Supplied on ROM with 76 page manual, and handy function key strip.

NEW

PRINTWISE — professional looking documents with the minimum of fuss

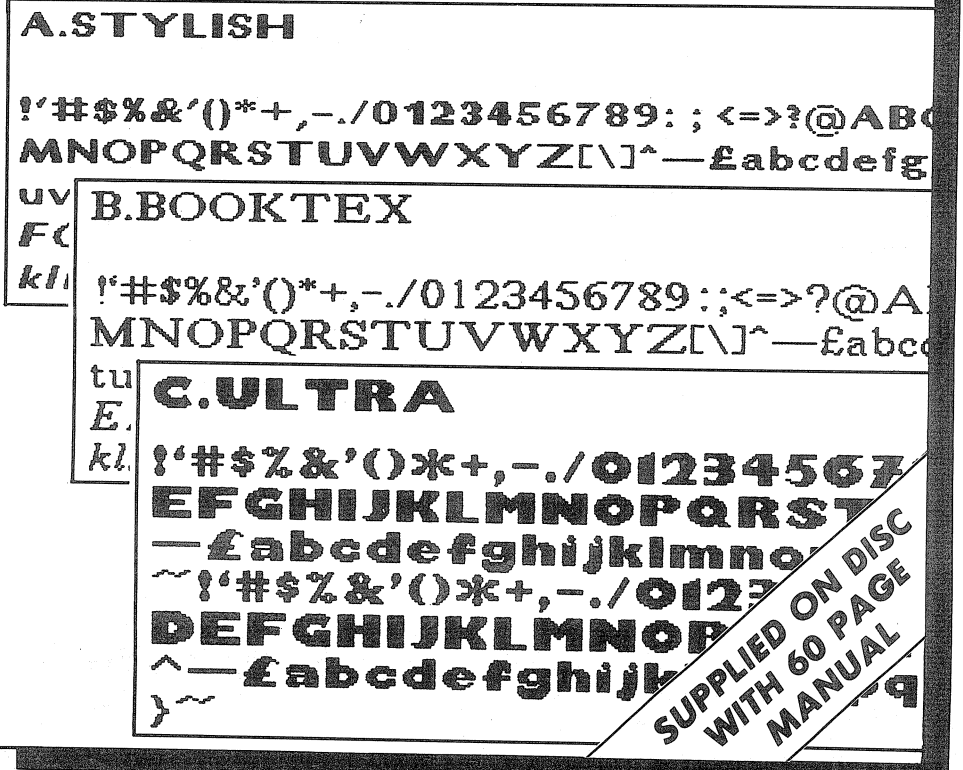
The ideal companion for Wordwise, View and Inter-word

Printwise is a low cost publishing aid, and allows you to create professional looking magazines, leaflets, posters — the possibilities are endless. Also use it for 'near-letter quality' printout for correspondence. Simply take your text file, specify the font styles you require, and Printwise will do the rest.

No programming skills are necessary.

PRICE	MEMBERS PRICE
£30.00	£22.50

- 18 authentic fonts covering 4 standard typesizes.
- all fonts can be used in the same document, or even in the same line!
- italic, bold and condensed typestyles.
- proportional spacing, right justification, full indentation.
- powerful font designer.
- handles any length text files.
- suitable for all Epson compatible printers, and Shinwa CP80.



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 ad if necessary. Any ads that 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 30p per word (inclusive of VAT) and these will be featured separately. Please send all ads (personal and business) to MEMBERS' ADS, BEEBUG, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX. The normal copy date for receipt of all ads will be the 15th of each month.

Tandy CGP115 printer plotter, hardly used includes BBC cable, screen dump program, new pens, paper £69. Tel (0983) 68861 after 6pm.

ROMs: Wordwise Plus, Ultracalc, StarDatabase; Disc: Beebplot (Gemini) - unused. Addison Wesley lightpen and software; Books: Disc systems for the BBC micro by Ian Sinclair, BBC micro ROM book by Bruce Smith, Making Music by Ian Waugh, Expert Guide by Mike James, Introducing the BBC by Ian Sinclair; Magazines: Micro User, Acorn User and A&B Computing. Any offers considered. Please phone Jennifer on (0908) 679349.

Wordwise Plus, complete with manuals £25. Amdek video - 300, high resolution green monochrome monitor with anti-glare screen £75. Tel: Rugby (0788) 813948.

Beebugsoft Toolkit ROM, unused, recent version, boxed with instructions £13. Phone (0786) 61501 evenings.

Watford CS400S cased single, 400K disc drive, 40/80 switchable with PSU and cable. Unused - £110. Elite on disc £5. Tel: Berkhamsted (04427) 6048.

Toolkit Plus ROM with manual, as supplied from Beebugsoft £20. Tel: Chorley (02572) 78286.

Wanted, Acorn Teletext receiver, with ROM and manual. Please phone Andy Fawkes, Broad Acres, Leckhampstead, Newbury, Berks RG16 8QY. Tel:(04882) 369.

6502 Second Processor £120. BCPL + calculations + generator £100. Acornsoft Forth + book £10. Complete BEEBUG vols 1-4 in binders £25. Various books and software originals at half price or less. Tel: Caxton (09544) 532.

6502 Second Processor £100. ATPL ROM board £20. Voltmace 14b joystick plus adaptor and disc £15. Disc Doctor ROM £15. Tel: (0963) 32131.

Following upgrade to Compact - For sale: Acorn Z80 Processor and all manuals and 2 sets of discs £275 ono; twin 80 track disc drive with own PSU £100 ono; also many ROMs at £10 each, and original disc games (inc. Elite) at £5 each. Tel: Roger Keyworth on 01-953 2214 evenings.

Various BBC 'B' programs for sale, tape/ROM/disc. Send S.A.E to Tony Jackson, 'Lawley', 26 Anson Road, Exmouth, Devon EX8 4NY.

BBC B, Acorn DFS, ATPL Sidewise ROM board, 16K sideways RAM, Beebugsoft Toolkit Plus and Sleuth ROMs - £250 ono. Will split. Tel: (0302) 744005.

Sleuth ROM and manual as new. £15 including carriage. Tel: 051-677 1518.

Star DP510 dot matrix printer (100 cps), tractor unit, cable and US manual. In regular use, good condition £95 ono. Tel: Hammersmith 01-741 0253.

Nightingale modem with autodial and Master compatible Commstar £110. Wordwise Plus £30. Masterfile II £10. 6502 Second Processor £135. All as new. Would like to swap a 'mouse' for a Marconi RB2 tracker ball in good condition (software not necessary). Tel: Paul on (0782) 815927 evenings, weekends.

Centronics GLP printer with tractor feed and cover £80 only. Tel: (0706) 73153 evenings only.

→34

Watford DFS 1.43 ROM, 10 chips, instruction manual. Half price - £34. Tel: 01-836 7640.

BBC B, Acorn DFS, 800K dual disc drives (40/80), Juki 6100 with tractor feed, amber monitor, Wordwise Plus and 4 Acornsoft business package programs. £595 for quick sale (no offers). Would consider hardware split. Tel: Barry Reed 01-669 2127 9am to 5pm.

Cumana CS100 40 track disc drive with 10 discs, manual, original packing, little used. MUST SELL £70 ono. Also tape Elite, Frak, Felix and others (won't run on my Master) going cheap. Tel: (0273) 832548 evenings.

Microline 80, 16cps, draft, enlarge, condense etc., vgc. Also 200K twin drive for sale. £75 for both, will split. Please contact Dominic (0785) 42483 or 5 Corporation Street, Stafford ST16 3LZ.

Acorn Z80 Second Processor including all CP/M packaged software, books plus Wordstar and dBase II £245 ono. Tel: Bedford (0234) 67067 evenings.

ROMs: Beebugsoft Murom, Toolkit and Sleuth, £12. Altra ROMs Enigma Disc Imager £15. Computer Concepts Disc Doctor £12. Tel: (0303) 42540 evenings.

6502 Second Processor £90. TRS80 line printer II £50. D/S 40 track disc drive £50. CGP115 colour printer £20. Monochrome monitor £20. Watford expansion board £10. TRS80 32K keyboard free with a purchase of any two items. All with ROMs and manuals. Tel: Bampton (0398) 31899 evenings.

Tandy CGP115 colour printer, little used, cable, power supply, 4 black pens £55. Tel: (0903) 892682.

For Sale: Wordwise Plus ROM, all manuals and cassette typing tutor £25. Gemini Office Mate (40 track disc) £6. Mini Office I (40 track disc) £4. Acornsoft Desk Diary and Address Book (tape) £4. Beebugsoft Design (tape) £4. All originals with instructions. Will separate. Tel: 051-724 3634.

BBC disc interface kit, including Intel 8271, DNFS and 9 other chips £48 ono. Tel: Bedford (0234) 67067 evenings.

Viewstore, new and unused £30. Mike Taylor, 7 Brockhurst Road, Gosport, Hants PO12 3AL.

BBC Master 128. Bought July 1986 £425 ono. Acorn 6502 Second Processor £125. Wordwise ROM £15. Tel: Stephen 061-881 1850 9am-6pm.

Owl, Beebon and other unusual BBC magazines wanted. Details to: Mr.Watkins, 101 St. Nicolas Park Drive, Nuneaton, Warwickshire CV11 6DZ.

Viglen professional console unit for BBC B £25. Also Viglen ROM cartridge system (no cartridges) £7. All perfect condition. Tel: Southampton (0703) 845104.

BBC B, Watford 2x800K plus PSU, Le Modem, Solidisk DDFS, SWR32 RAM, Quinkey, monochrome monitor, Viewsheet, Intersheet, 3 years BEEBUG, Acorn User. Offers - (0344) 884309 evenings/ weekends.

Wanted: Aries B32 shadow RAM and B12 ROM boards. Sensibly priced items please. Tel: Ripon (0765) 4613.

12 multistrike film ribbons, black (suitable Qume IBM6240). 2 Diablo French Prestige Cubic printwheels No.38131-01. 1 Bi-lingual Courier printwheel No.M2083. All new and unused. Qume S3/45 daisywheel printer (parallel 13 bit Wordplex interface) plus power unit model 530. Offers invited for above items. Acornsoft View guide £3.50, Viewsheet user guide £6.50. Tel: Terry (0705) 553546.

3 inch Hitachi disc drive 40 track complete with some discs and software £50. Tel: Chester 310265.

Phloopy drive wanted in working order. Offer to 63 Inverness Street, Sunderland.

For Sale: BBC 'B' OS 1.2, Solidisk DD DFS. Twin 40/80 D/S drives, Wordwise Plus, Masterfile II, 20+ discs, mounted in Solidisk cabinet. Games, books, Manuals. Suggest £300 ono. Tel: (0202) 432973.

Wanted: Teletext Adaptor. Tel: David (0494) 21588 office hours.

BEEBUG

Stand 13

(Hall A Lower)

Exhibition Opening Times

Wednesday 21 January	10.00-18.00
Thursday 22 January	10.00-18.00
Friday 23 January	10.00-18.00
Saturday 24 January	10.00-16.30

A complimentary ticket for this exhibition is enclosed with your magazine.

SEE US AT

THE THIRD
HIGH TECHNOLOGY
AND
EQUIPMENT
IN
EDUCATION

Exhibition
E87

January 21st-24th 1987
BARBICAN, LONDON

Business Ads

MARKS AND STATISTICS For teachers and lecturers. Simple to use spreadsheet format. 330 students per file. Up to 300 subjects. Sorts, analyses, normalises, calculates means, applies weighting factors. Prints histograms and lists. Full documentation supplied. Disc (40 or 80 track) £17. In-Form, 73 Woodfield Park, Colinton, Edinburgh EH13 0RA.

Rural Rambles and Seaside Strolls - new versions of General Map Reading and Coastal Map Reading (see Memory Maps, BEEBUG July 1986) make map reading an exciting game. BBC B/B+/Master, disc only, £9.95 each plus £1 p&p. Soft-Teach, Longbridge Deverill, Warminster, Wilts. BA12 7EA.

Fed up with zapping aliens? How about a game to tax your brain? At only £4.95 tape, £5.95 disc, Sharemaster offers you the chance to make thousands on the stock market, from your armchair. Cheques/cash/P.Os to John Adams, Buckland House, East End Paglesham, Rochford, Essex SS4 2EQ. (B, B+ and Master).

Clipart. Digitized pictures, cartoons and silhouettes available in mode 0 as cutouts for Pagemaker or in mode 4. 80 track disc with 80+ pictures £5.80. 40 track available on 2 discs £9.60. Your own pictures digitized. Parkside Developments, 9 Ingleby Gardens, Chigwell, Essex IG7 6EH. Tel: 01-500 5701.

Events

Higher Technology and Equipment in Education	Barbican Centre London	21-24 Jan
Electron and BBC Micro User Show	Old Horticultural Halls, Westminster	20-22 Mar
Electron and BBC Micro User Show	Old Horticultural Halls, Westminster	8-10 May
PCW Show	Olympia, London	3- 8 Sep
Electron and BBC Micro User Show	Old Horticultural Halls, Westminster	13-15 Nov

Discounts Update

Akhter Computer Group have advised us that they are no longer able to offer discounts to BEEBUG members.

MASTER SERIES

Art Room Master

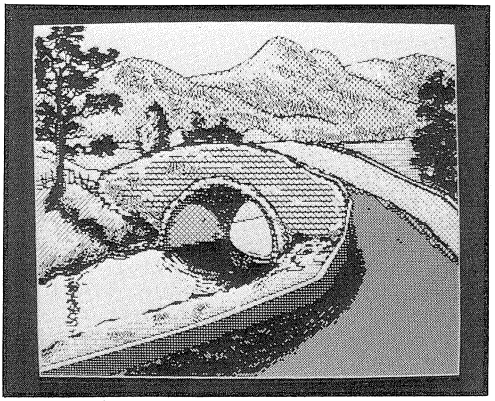
One of the first packages for Acorn's new Compact is Clare's Art Room, already selected by Olivetti for sale with the Italian version. Mike Williams paints the scene.

Product: Artroom
Supplier: Clares Micro Supplies, 98 Middlewich Rd, Northwich, Cheshire CW7 4AX. (0606) 48511
Price: £25 (Master) £27 (Compact) Demo Disc £2

Clare's Artroom must be one of the first products written specially for the Master Compact, though it is also available in ADFS format for the Master 128. Indeed, the program has been bought by Olivetti and is being bundled in as part of the software with the Prodest, Olivetti's repackaged version of the Compact now being sold in Italy. The Compact version comes on a 3.5" disc, and the Master version on the usual 5.25 floppy. Both versions are accompanied by a 44 page manual. The whole package is designed to be mouse-driven, as well as controlled from the keyboard, and Clares recommend the Nidd Valley mouse at £59.90.

Perhaps surprisingly, Artroom is in black and white. Clares claim that as the prime objective of the package is to produce high quality graphics for output to a printer, colour is unnecessary. The package also includes a 'Graphics Library' consisting of some 16 screens produced by Artroom. The manual implies that this 'extensive' library provides a wealth of graphics for the design of news sheets, posters, etc., and suggests that Artroom has some of the capabilities of the likes of Pagemaker and Fleetstreet Editor (written by Clares for Mirrorsoft). This is not really true. The collection of 'pictures' and images in the graphics library will, I imagine, provide only limited material for those designing their own graphics displays - as examples of what can be achieved with Artroom the library is excellent, as the accompanying illustrations show.

Having so far cut Artroom down to size, lets see just what it is capable of doing. When first booted, Artroom presents the user with two main choices, the creation and editing of graphics screens, or outputting screens to a printer. The choice is presented graphically, and I had some difficulty in working out which I had chosen - the manual is not too clear. Once into the main graphics option you are presented with a large icon menu in a matrix of five rows by eight columns. This occupies a central position on the screen and thus overlays any picture you are creating while selecting from this menu.



Within a short while, most of the icons become familiar, but two of them appear identical (!) and can only be distinguished by their position.

Probably the best way of getting started is by loading one of the pictures from the graphics library and experimenting with that. Clares have taken that to heart, providing a part finished picture and a complete 'guided tour' of the package based on this picture.

Basic drawing can be accomplished by a set of five 'pens' of different thicknesses. There is also an airbrush and a 'rubber band' form of drawing. With this, one point is fixed, and then moving a cursor about the screen causes a line to

be moved as though a rubber band were being stretched between the fixed point and the cursor.

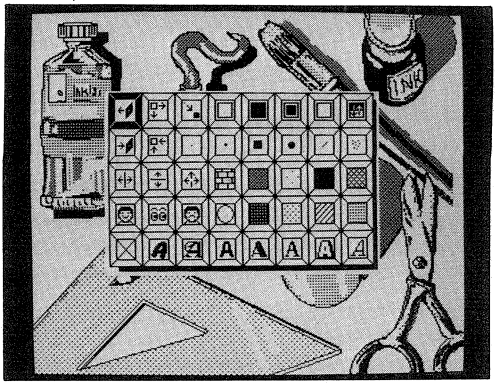
Four types of box can be drawn, outline or solid and black on white or white on black. In each case a standard rectangle is first positioned to fix the top left hand corner, and then enlarged to the required size and shape. Circles (but no other curves) can be similarly positioned and sized.

Nine different 'fill' patterns are provided, but although reasonably speedy, are somewhat erratic in execution. Many times I found quite simple areas only partly filled, and repeated application was needed to achieve what I had set out to do. The fill also has a tendency to 'leak' when least expected. The manual does warn of both of these situations, but the results should and surely could have been better.

One of the most useful and attractive features of the package is its ability to copy, and at the same time enlarge or reduce, selected areas of the screen. However, the area chosen can only be enlarged or reduced by a factor of two, though this can be done vertically, horizontally, or both. Again this is a limitation on the flexibility of the package. It is also possible to select reflection about either a vertical or horizontal axis at the time of enlargement giving further possibilities. As it is also possible to save and reload selected parts of a screen, it is feasible to build up a library of frequently used shapes and symbols, but it is up to the user to standardise on which part of the screen is to be used for this purpose. A symbol could then be loaded from your library, and copied to the required location.

A further excellent feature is the ability to zoom in on a part of the screen and deal with that section at pixel level. And it is easy to scroll, both vertically and horizontally when in this magnified mode, with a small normal-scale picture top left for reference.

Text can also be included in any display using one of seven fonts supplied (compatible with Clares' Fontwise), but only in one size. Once on the screen, text can be copied, enlarged or reduced as



already described for pictures. However, reducing the size of text can lead to very poor if not unreadable quality.

One final feature must be mentioned. Amongst the icons will be found a 'smiling face', a 'sad' face and a pair of eyes. The happy face allows the current display to be saved elsewhere in memory as a reserve screen, which can be restored with the sad face. The manual advises frequent back-ups of your work in this way so that any mistakes can be readily restored from a slightly earlier version. This is not totally under the control of the user as certain functions automatically up-date the reserve screen with the current picture before taking effect (particularly the fill function). The 'eyes' allow you to see what is in the reserve screen.

The printout facilities are quite straightforward, if you have an Epson compatible printer, providing control over both the screen area to be dumped and the position on the paper. You can also view a screen before it is dumped (as the dump works direct from disc).

CONCLUSIONS

Most beginners would find Artroom quite an enjoyable fun package, despite its limitations. However, there have been many better art and design packages for the BBC model B, many of which are compatible with or are being released for the Master and Compact. In my view, Olivetti have done the right thing in bundling this package in with the Prodest (Compact) for the first time user. Others, with more experience, will do better to look elsewhere.

David Graham presents four function key definitions to automate DFS/ADFS transfers.

Fiddling about with the parameters of the intricate *MOVE command can take forever. The function key definitions presented here solve the problem. The five key definitions perform the following processes:

f1 Catalogue DFS disc
f2 Load file from DFS disc
f3 Save file to DFS disc
f4 Copy file from DFS to ADFS disc
f5 Copy file from ADFS to DFS disc

The software is intended to operate from within the ADFS, and in all cases it is assumed that the ADFS disc is in drive 0, and that the DFS disc is in drive 1. It is packaged up as an EXEC file for ease of use, and is probably best typed in using the Editor. If the file is saved (using the SAVE option in the Editor) under the filename DISCKEYS, then typing *DISCKEYS in immediate mode will load the keys without disturbing any resident program.

As you can see, the EXEC file is self-documenting, with a legend displayed on the mode 0 screen. Mode 0 is used in preference to the shadow modes, because keys f4 and f5 use *MOVE which makes use of shadow memory as workspace. If you change to mode 128, say, and then try to use f4 or f5, your drives will shake to pieces and you will become old before the transfer is complete.

Keys f4 and f5 solve the problem of filename length and directory as follows. If you try to transfer an ADFS file with a filename of more than 7 characters, it is truncated to 7, and all transfers are put in the "A" directory on the DFS disc.

In the reverse direction, DFS filenames which include a directory letter (e.g. W.TXTFILE) are truncated up to the full stop, and all transferred files are saved in the currently selected ADFS directory.

One or two tricks have also been employed for blanking the screen, and so on, and these are discussed more fully in the accompanying EXEC Files Update article. But note that f9 is also defined. This key is called by key f5 because the program defined in key f5 is too long for a single key. Key f9 can in fact be used on its own. Its function is to unlock all DFS files. This is necessary because *MOVE leaves all transferred files (DFS and ADFS) completely locked.

```
MODE 0
*****************************************
*|      DISC - ADFS TRANSFER  ver 8      *
*|             FUNCTION KEYS             *
*|                                       *
*| Key 1 : Cat DFS                       *
*| Key 2 : Load file from DFS            *
*| Key 3 : Save file to DFS              *
*| Key 4 : Copy DFS file to ADFS (*MOVE) *
*| Key 5 : Copy ADFS file to DFS (*MOVE) *
*|                                       *
*****************************************
CO.0
*KEY1 *CAT -DISC-:1|M

*KEY2 CO.0|MCO.7:SO.1,-15,200,2:INPUT"LOAD
 filename "A$:OS.("KEY0 LOAD "+CHR$34+"-DI
SC-:1."+CHR$34+"+A$"+CHR$124+"M"):*FX138,0
,128|M

*KEY3 CO.0|MCO.7:SO.1,-15,200,2:INPUT"SAVE
 filename "A$:OS.("KEY0 SAVE "+CHR$34+"-DI
SC-:1."+CHR$34+"+A$"+CHR$124+"M"):*FX138,0
,128|M

*KEY4 CO.0|M:CO.7:SO.1,-15,200,2:INPUT"COP
Y DFS to ADFS Filename "A$:Z$="MOVE -DISC-
:1."+A$+" -ADFS-@.":IFMID$(A$,2,1)="." A$=
RIGHT$(A$,LEN(A$)-2):OS.(Z$+A$):OS.("A. *
WR") ELSE OS.(Z$+A$):OS.("A. * WR")|M

*KEY5 CO.0|M:CO.7:SO.1,-15,200,2:INPUT"COP
Y ADFS to DFS Filename "A$:Z$="MOVE -ADFS-
@."+A$+" -DISC-:1.A.":IFLEN(A$)>7 A$=LEFT$
(A$,7):OS.(Z$+A$):OS.("FX138,0,137") ELSE
OS.(Z$+A$):OS.("FX138,0,137")|M

*KEY9 CO.0|MCO.7:OS.("disc"):OS.("DR.1"):O
S.("A. A.*"):OS.("ADFS")|M

P.STRING$(14,CHR$11):SOUND1,-15,50,2:CO.7
```

MASTER SERIES

Master Hints

More hints for the Master and Compact from David Graham.

Keypad Negative INKEY

The negative INKEY values for the keys on the Master keypad are different from those on the main keyboard. For those without Reference Manual part one (D.2-39), here they are; and for those with it, note the value for the decimal point:

0	-107	5	-124	+	-59
1	-108	6	-27	-	-60
2	-125	7	-28	/	-75
3	-109	8	-43	#	-91
4	-123	9	-44	*	-92
,	-93				
.	-77				
Del	-76				
Ret	-61				

W.R.Davis

Shady Moves

*MOVE is a powerful, if ponderous, command that is invaluable for transferring files between DFS and ADFS. It cleverly uses shadow memory as workspace. This means that, unlike *COPY, it does not corrupt resident programs. But BEWARE, if you are in a high resolution shadow mode when you use it, it will take forever to transfer a file of any length, and your drives will be reduced to rubble in the process. (He exaggerates - Ed.)

Editor Search and Replace

The Master Editor contains a very powerful group of search options, though you need to study the Reference Manual part two with some care and a little scepticism before you can get the best from it. Here are three useful examples.

```
f5  $Return
f5  $^Space^#\:/Return
f5  ^#REM*~$$/Return
```

The first will count the number of lines in a Basic program, without the need to renumber it. The second will remove all Basic program lines consisting of a single colon, and the third (unlike the Reference Guide version) will remove all Basic lines which start with a REM.

Default Baud Rates

The default Baud rates on the Master's RS423 port are different from those on the standard model B, though of course you can reset the default rates on the Master with *CONFIGURE. As shipped, they are set to 9600 on the B and 1200 on the Master.

DATA Statement Quirk Fixed?

If you run the following program:

```
10 RESTORE10:READvar:PRINTvar:DATA 1,2
20 DATA 100,200
```

the Master will print 100, and not 1 as might be expected. In other words, DATA statements may not be placed on the same Basic line as the READ statement. This is different in Basics I and II on the model B, which would print 2 as the result of the same program, since they just lose the first piece of data on the line.

WYSIWYG for Wordwise

Master users who are also Wordwise devotees have a neat way to obtain WYSIWYG, something particularly handy for setting out tables, which can be a pain in the eye with Wordwise. The solution is to use the mighty Editor to create the table. This gives full screen 80 column layout. Once the table or whatever is produced, just save it away using f3, and load it into Wordwise using option 2 or 4. It needs no editing to remove codes etc, but if you need to edit it for other reasons, it may be easiest to use the Editor again.

If you enter Wordwise after visiting the Editor, you will find that the cursor keys do not behave with Shift and Ctrl quite as they should. This is fixed by pressing Break from the Wordwise menu, or from Basic after leaving the Editor.

Bad RENAME

When renaming directories, you cannot use the root specification ($). Thus the following:

```
*REN. $.FIRSTNAME $.SECONDNAME
```

will give a "Bad rename" error. The only way around this is to select the root directory as the current directory, and use:

```
*REN. FIRSTNAME SECONDNAME
```

David Graham reveals some tricks which make EXEC files behave even more like MS-DOS batch files.

A couple of issues ago (BEEBUG Vol.5 No.6) we looked at an extremely powerful way of using EXEC files to perform a whole host of tasks on the Master and Compact. The idea essentially involved using EXEC files much as batch files are used on the IBM PC. There are however a number of commands provided by MS-DOS that Master batch files lack. But with a few tricks we can bridge the gap.

SCREEN BLANKING

When you call a complex EXEC file, the whole contents of the file are displayed on the screen. This is messy, and obscures any legitimate output that the file might produce. PC users have the three neat commands ECHO, ECHO ON and ECHO OFF to control "echo" of file contents and messages to the screen. The best way to achieve a similar result in BBC Basic is to set the display mode early in the file to any mode other than 7 or 135, and to use COLOUR Ø (CO.Ø) to turn off the display and COLOUR 7 (CO.7) to turn it back on again.

This technique is used in the EXEC file in the DFS-ADFS Transfer article elsewhere in this issue. CO.Ø is used to turn off all subsequent display after the user message is put up. The same command is also used within the function key definitions set up by the EXEC file. Notice that in key definitions f2 to f5, CO.Ø is followed by |M (to generate a Return). This activates the command immediately. You will notice that it is directly followed by CO.7. This is to ensure that when the *INPUT* statement is executed, any accompanying message and the user's response will both be visible.

The last line of the file is also worthy of note. The instruction P.STRING$(14,CHR$11) prints a string made up of a series of ASCII 11s. These take the cursor back up the screen, and thus get rid of the gap left by the invisibly printed text. The CO.7 at the very end then reinstates normal text printing.

USER INPUT

There is a basic problem with all user input to an EXEC file, as you can see if you create the following EXEC file:

```
INPUT"Go ahead (Y/N)";ans$
IF ans$="Y" THEN P."Action Required"
```

Where things go wrong is that since the EXEC file simulates keyboard input, whatever follows the INPUT statement in the file, is itself taken as user input. In other words, the computer takes the line following the INPUT statement to be the user's response. The same thing happens with GET and INKEY. But there are ways around the problem.

The simplest involves using negative INKEY to detect a key-press, and this works because the negative INKEY function scans the keyboard directly, rather than using the keyboard buffer (which is where the EXEC file contents are directed). Thus you could use the following:

```
P."Go ahead?";:REPEAT UNTIL INKEY(-69)
REM Flush buffer
*FX15,1
REM The rest of the file
```

The top line will now wait until the "Y" key is pressed, and will hold up the EXEC file until this is done. If you wish to abort the file, then just press Escape instead of "Y". The purpose of the FX15 is to flush the keyboard buffer of your (now redundant) answer.

Although you cannot test a whole range of keys with negative INKEY, you can take this a little further. For example, the following is a little more elegant:

```
P."Go ahead (Y/N) ?";:REPEAT UNTIL INK
EY(-69) OR INKEY(-86)
IF INKEY(-86) THEN *FX125
*FX15,1
REM The rest of the file
```

Here, pressing "N" will cause the file to abort, since FX125 simulates an Escape.

Sometimes however, you need to use an INPUT statement. This is possible from within an EXEC file, but there must be no characters following the first Return character after the INPUT statement. If there are, then these will again be taken as user input. This can be quite restricting, since no EXEC file line may be more than 255 characters in length. If you wish to go over this length, you can

either use a function key or call in a second EXEC file.

EXEC files may be chained one after the other simply by inserting a command to EXEC in the next file, at the end of the previous one. The command should be at the end of the file, since calling a second EXEC file cancels the first. Calling a function key from an EXEC file is also quite easy. You need to set up the contents of the key early in your EXEC file using *KEY, and call it with the following FX call:

```
*FX138,0,n
```

where n is a number which defines which key is called. 128 calls key f0, 129, key f1, and so on. It is important to note that although you may place this FX call early in an EXEC file, it will not be implemented until the end of the file is reached. As an example, the following brief EXEC file sets up key f9, and then calls it:

```
MO.0:CO.0
*KEY9 CO.0|MCO.7:P."This can follow an
 INPUT statement"|M
CO.7:INPUT"Filename "f$:*FX138,0,137
```

Although this example is quite artificial, it does show that the text printed by the function key does not upset the INPUT statement in the EXEC file executed earlier. You may have noticed that the DFS-ADFS transfer EXEC file mentioned above uses a similar technique in key f5 to call key f9. But this is just because the program is too long to get into a single key definition.

REPLACING GOTO

EXEC files have no line numbers of course, so it is not possible to implement GOTOs of any kind. But you can achieve program branching with extensive use of the IF THEN ELSE construct. If you have more than one branch that needs to be more than one program line in length, you will need to use more than one IF statement; and in this case it will be as well to reduce this to simply testing the state of a single variable, or a flag. It is also possible to set up and call procedures and functions from EXEC files, as an alternative to using GOTO, but I leave it to you to look for the most convenient way to implement this.

MASTER SERIES

Maintaining Status

David Graham shows how to save your CMOS RAM settings to disc as an emergency back-up.

When your Master's backup battery goes flat, you will lose all your machine status settings. This short program allows you to save them to disc (before your battery goes down), and load them in again whenever you need to. It is also handy if you wish to use different status settings at different times.

When you run the program, it asks if you wish to Read or Write the CMOS RAM. If you select Read it will read the RAM, display the contents on screen, and save it on file under the name "CMOS". If you select Write, it will look for a file of the same name, and both display its contents, and write it back to the RAM. You can test it out using *STATUS and *CONFIGURE.

```
  10 REM CMOS RAM READ/WRITE ver8
  20 PRINT'"Read or Write RAM (R/W) ?";
  30 CLOSE#0:A=GET:PRINTCHR$A
  40 IF A=82 THEN PROCread
  50 IF A=87 THEN PROCwrite:PRINT"New s
ettings take effect after next Break or
power-up"
  60 CLOSE#0:END
  70 :
1000 DEFPROCread
1010 A%=161:B%=OPENOUT("CMOS")
1020 FOR X%=1 TO 49
1030 Z%=(USR(&FFF4)AND&FF0000)DIV&10000
1040 PRINTX%,Z%:BPUT#B%,Z%
1050 NEXT:ENDPROC
1060 :
1070 DEFPROCwrite
1080 B%=OPENIN("CMOS")
1090 FOR X%=1 TO 49
1100 Z%=BGET#B%:PRINTX%,Z%
1110 OSCLI("FX162,"+STR$X%+","+STR$Z%)
1120 NEXT:ENDPROC
```

Sorting Data Files (Part 2)

This month, Jan Stuurman shows in detail how the sort program presented last month can be readily adapted to work with different data files.

We will now look in more detail at the implementation of the techniques for file sorting that we examined in principle last month, and show how the Filer Sort program can be adapted to sort other data files. If you remember, the basic idea for file sorting was as follows:

1. Determine the composition of the 'key field' (or sort field) which will be used as the basis of the sort.

2. Read each record in turn, extracting the key field and saving this in memory with a simple pointer (or tag) to the full record on disc.

3. Sort the key fields (and tags) in memory using the Shell sort technique.

4. Use the re-ordered tag file in memory to read each record from the original file and write these out to a new file in the new order.

All line number references are to the listing given last month. The two arrays field$() and width%() are used to hold the names and widths of the fields that make up each record (a fixed field length, fixed record length format is assumed). For Filer, the information to be stored in these arrays is read (PROCopen at line 8000) from the File Description Record at the start of the file, but you will have to decide for yourself how to set up these arrays for other files.

In the program the first step is to determine the file to be sorted, and the new file to be created. The information is collected by the procedure PROCfilenames (at line 4000), and the names of the two files assigned to the variables op$ and np$ for use later. Two further procedures called PROCopen and PROCnopen are used in the Filer version to read and write the File Description Record. This is where the arrays field$() and width%() are filled.

The sort program requires three further arrays:

sfld% field number (position in record)
strt% start character in field
len% number of characters from field

These arrays are used to build up a picture (or template) of how fields, or parts of fields, are to be combined to form the key field for each record. This process is handled by PROCsortfield (line 5000 onwards). For each field (or part field) used to form the sort key, these arrays store the field number, the position of the first character in the field to be used (default 1), and the number of characters from the field (default from the first character to the end). The maximum numbers of fields per record, and fields (or part fields) per sort key are determined by the sizes with which these two sets of arrays are dimensioned (lines 1030 and 1040).

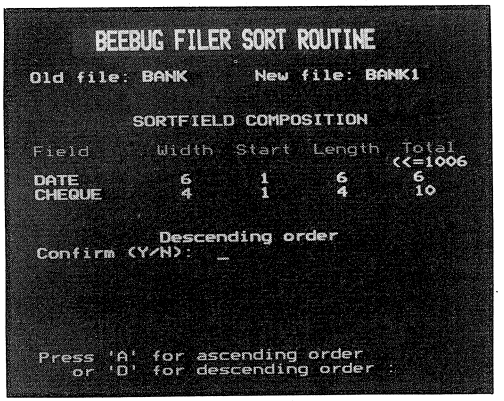
The two procedures PROCalfnum and PROCascdec are called to determine whether a numeric or alpha sort is required, and whether to sort in ascending or descending order. The variables t$ and o$ are set as required (lines 6050 and 7050).

Once the key field and other information has been determined, the procedure PROCsort (line 11000 onwards) handles the remaining stages. The coding here is quite difficult to follow, but exceedingly neat in the techniques used. Line 11030 uses the information now stored in the three arrays sfld%(), strt%() and len%() to build up a complete string (s$) of Basic instructions as a template or mask, which can then be EVALuated to extract and combine the relevant parts of each record as it is read in.

At the start of the main program, the value of HIMEM is set to be &1800 higher than PAGE. The program (as written) fits below the new value of HIMEM, and the memory above up to the start of the mode 7 screen is used for storing the tag file. The start address of this memory area (value of HIMEM) is assigned to B% (1020).

The records are read in one by one, the key field extracted using the template s$, and the key field and tag (or pointer) are store in memory from B% upwards (lines 11050 to 11070). L% holds the length (number of bytes) of the combined key field and tag.

The sort involves comparing keys in memory and swapping as necessary to produce a re-ordered tag file. The comparison used is again built up as a string of Basic instructions (stored in t$) and EVALuated at line 11120 to sort the tag file. Once the sorting is complete, records are read again from the original file (using direct access via the tag) and written out sequentially to the new file.



CREATING YOUR OWN PROGRAM

If you have read the above description carefully, referring to the listing of the program given last month, you should now be able to amend the program to sort other data files. The essential procedures and functions for your own version of the sort program are as follows:

PROCsortfield	PROCsort
PROCalfnum	FNfwidth
PROCascdec	PROCwindow1
PROCwindow2	

You will also need your own versions of the procedures PROCread and PROCwrite to read and write a record (from or to disc) using the array record$() as a buffer to hold one record at a time. In practice, you may well find it easier to work from the original program, modifying this as required rather than starting from scratch and incorporating the procedures needed. All the input required from the user is dealt with quite conveniently by PROCinput (at line 3000).

Other procedures and functions used:

ask	gets Y/N response
close	closes both old and new files
error	provides error handling
filenames	gets details of file names
input	collects all input required
nopen	opens new file
open	opens old file
oscli	implements OSCLI
read	read a record
setup	initialisation
strip	strips trailing characters
title	displays title screen
write	write record to file

The procedures and functions PROCopen, PROCnopen, PROCread, PROCwrite, and FNstrip are specifically related to the Filer data file structure, and if used, will need amending to suit your own data files. The other procedures and functions listed above are not related directly to Filer, but remember that they may in turn call other procedures which are.

More important variables:

rec	number of records in file + 1
R	position of a field in a record
F1	channel of old file
F2	channel of new file
len$	length of field on input
lf	length of sort field
maxl	maximum length for sort field - (see line 8040)
np$/op$	old and new file names
o$	ascending/descending character
s$	sort field template
t$	sort comparison
sf	sort field number
sfld$	key field name on input
strt$	start character on input

Clearly, the task of amending a program like the Filer Sort is not for the beginner, but the readable structure of the program, together with the description of its use last month and the program notes given here, should should go a long way. For convenience, the Filer Sort program is repeated on this month's magazine cassette/disc.

Why use error traps in your programs? C.P. Yu explains.

The BBC Micro has a perfectly adequate way of handling errors at run-time. If you try the following simple program:

```
10 WOOPS!
```

you will see the machine acquit itself impeccably with the reasonably useful message:

```
'Mistake at
 line 10'
```

So why should we need to incorporate error handling routines, and what is the purpose of the armoury of error handling equipment (ERR, ERL, REPORT and so on) at the disposal of the BBC Basic programmer?

Well, I can think of three good reasons for their use, and there are probably many more. Firstly, an error trap allows you to reset the machine to any desired state on pressing Escape or on encountering any other run-time error. This is very useful if your program disables the cursor keys or function keys, or leaves you in modes 2 or 5, or if it leaves disc files open.

Error handling routines can also allow you to use the Escape key to take you to any desired place in your program, such as a main menu, or even to allow the Escape key to toggle between two different screens, as it does in Wordwise or View (though these are of course both machine code programs). Finally, error handling routines allow you to filter out different kinds of errors, so that your program can react differently when it encounters a division by zero error to when it tries to write to a disc with a write-protect tab.

THE EQUIPMENT

All this can be achieved using the four error handling constructs:

ON ERROR, ERR, ERL and REPORT

The first of these, ON ERROR, is used to trap the error and direct the program to the user's error handling routine. This would take the form:

```
ON ERROR GOTO 1000 etc
```

A variant of this, ON ERROR OFF, is used to switch off user error handling, and to revert to the Beeb's own built-in error handling software.

ERR and ERL are two special functions which return respectively the error number of the last error, and the line number at which it occurred. REPORT, by contrast, actually displays the error message associated with the last encountered error. And somewhat idiosyncratically, if there has been no error since the machine was powered up, REPORT prints the Basic copyright message.

PUTTING IT TOGETHER

With these building blocks we can easily build useful error handlers. Here is one of the simplest:

```
100 ON ERROR GOTO 5000
...
Main program
...
5000 REM ERROR HANDLER
5010 ON ERROR OFF
5020 MODE 7
5030 REPORT:PRINT" at line ";ERL
5040 END
```

The ON ERROR statement should generally come as early in the program as possible, so that even the earliest errors are directed through the routine. Note here the use of ON ERROR OFF. This turns off the error handler once it has been entered, so that should there be any error in the handler itself, you will not be stuck in an endless loop, but instead, the error will be reported by the machine's internal error handling software.

Line 5020 puts the machine into mode 7. This is useful because it clears the screen, re-enables the cursor in case it was disabled, and puts the machine in a readable text mode. Line 5030 then REPORTs the error message, and prints up the line number at which it occurred.

If your program disables the function keys or cursor keys, then you should add

*FX225,1 to reinstate the former, and *FX4 the latter; and if it opens disc files at any point, you should include CLOSE #0 to close them. VDU3 is also often included to switch off the printer should an error occur while the printer is enabled. Incorporating these, we get the following:

```
100 ON ERROR GOTO 5000
...
Main program
...
5000 REM ERROR HANDLER
5010 ON ERROR OFF
5020 MODE 7
5030 REPORT:PRINT" at line ";ERL
5040 *FX4
5050 *FX225,1
5060 CLOSE#0:VDU3
5070 END
```

There are other things which may also need to be reset, such as cursor flash rates, keyboard auto-repeat rates, and so on; but if you are writing the program which unsets these parameters, you should have little difficulty in resetting them in the error handler.

SIMPLE ERROR FILTERING

The ERR function allows us to filter errors in such a way that a program reacts differently to different kinds of error. The most commonly used filter is probably that used to respond to the Escape key. When you press the Escape key at any time, it generates an error just as would happen if the error had been inherent in the program itself. Pressing the Escape key generates error number 17, and we can use this information to respond to Escape in any way that we choose.

For example, if we insert the following line:

```
5005 IF ERR=17 THEN GOTO 500
```

we can ensure that pressing Escape always takes us to line 500. This could be the start of a menu screen or whatever. If we are going to use this technique, there are two things to bear in mind. The first is that we cannot redirect the program to ANY point that we choose (currently line 500). This is because when an error is encountered, Basic clears the 6502 stack on which are stored essential loop data. The result of this is that once an error is encountered, the computer loses all "memory" of its current position in FOR-NEXT and REPEAT-UNTIL loops. We must therefore ensure that no error handling routine directs the program to the inside of any loop.

ESCAPE TO EXIT

The second problem is that using Escape in this way does not allow us to use it to actually ESCAPE from the program, since every time that we press Escape, we are taken back to line 500 or whatever. The way around this is to have a second ON ERROR statement active during the menu screen only, which allows error number 17 (i.e. Escape) to exit from the program. This might work as follows:

```
100 ON ERROR GOTO 5000
...
200 PROCsetup
500 REPEAT
510   PROCmenu
520   ON ERROR GOTO 5020
530   REPEAT
540     A=GET-48
550   UNTIL A>0 AND A<9
560   ON ERROR GOTO 5000
570   IF A=1 PROCfirstchoice
etc
600 UNTIL FALSE
...
5000 REM ERROR HANDLER
5010 IF ERR=17 THEN GOTO 500
5020 ON ERROR OFF
5030 MODE 7
5040 REPORT:PRINT" at line ";ERL
5050 *FX4
5060 *FX225,1
5070 CLOSE#0
5080 END
```

In assessing the way in which program segments such as the above will function, you need to keep a close watch on the sequence of ON ERROR statements. In this case the first is encountered at line 100. Any error occurring after line 100 directs the program to the handler at line 5000. This distinguishes between Escape (error number 17) and all other errors. If the error number is 17, the program is directed to the start of the menu sequence at line 500. Note that this line is outside ALL program loops. Any other error number causes the program to stop, with the appropriate error message, handled by lines 5020 to 5080.

Once the program enters the menu sequence, and halts at line 540 waiting for the user's choice, the ON ERROR

destination line number has been altered slightly. This was achieved in line 520 just before the GET statement. If any error is now encountered, the error handler directs the program to line 5020, thus bypassing the Escape filter, so that if Escape were pressed from this point, the program would no longer return to the menu (which would have been pointless, since you are already in the menu). Instead the program terminates gracefully through the error handler (lines 5020 - 5080). Thus we have achieved our objective: pressing Escape from within the menu terminates the program, while pressing Escape from anywhere else takes you to the menu.

AUTO ERROR LISTING

Next month I will be continuing with this theme, looking firstly at more complex uses of the Escape key, and secondly at more involved error filtering. But by way of light relief, the remainder of this article is devoted to automatic error listing, and techniques for disabling the Escape key.

With a little persuasion, the computer can be made to list automatically the line in which an error occurred, rather than just giving us its line number. The reason why persuasion is called for at all is that the command LIST cannot normally be issued from within a Basic program. If it could, we could just include a line such as:

```
1000 LIST ERL or whatever
```

One way around the problem is to call the LIST command from within a function key definition, and then use an FX call (FX138) to simulate the pressing of the key. The following error handler uses this approach.

```
  10 ON ERROR GOTO 1000
  20 WOOPS
1000 ON ERROR OFF
1010 MODE7:REPORT:PRINT
1020 $&900="K.0L."+STR$(ERL)+"|M":X%=0:
Y%=9:CALL&FFF7
1030 *FX138,0,128
1040 END
```

Line 1010 of this routine uses REPORT to describe the error type, then the complex code in the following line sets up function key zero with the immediate command to list the error line. Line 1020 is made considerably more complex by the necessity of having to work on Basic I. If you have Basic II, or a Master or Compact, you could replace line 1020 with:

```
OSCLI("K.0L."+STR$(ERL)+"|M")
```

Line 1030 uses FX138 to simulate the pressing of function key zero, into which the LIST instruction has been stored. You may well find it useful to include these lines in programs under development, since the listed line may easily be cursor-edited, to correct the error.

Another way to list automatically an error line is to use the function key directly. In other words, when you hit an error, you press f0 or whatever, and the line is listed to the screen. The advantage of this approach is that it requires no adjustment to the program itself. Listed below is the key definition which achieves this:

```
10 *KEY0 A$="L."+STR$(ERL)+CHR$13:A%=
138:X%=0:F.A=1TOLENA$:Y%=ASC(MID$(A$,A))
:CA.&FFF4:N.|M
```

ESCAPE FX CALLS

Finally I want to mention a number of FX calls related to the use of the Escape key. FX200 may be used to completely disable the Escape key. Its variants are as follows:

*FX200,0	Enable Escape Key
*FX200,1	Disable Escape Key
*FX200,2	Enable Escape Key, and clear memory on Break
*FX200,3	Disable Escape Key, and clear memory on Break

The other call is used to REDEFINE rather than DISABLE the Escape key:

```
*FX220,n
```

This call redefines the Escape key to be the key whose ASCII value is n. Thus:

```
*FX220,0
```

makes the sequence Ctrl-@ generate an Escape, since the ASCII value of Ctrl-@ is 0. This particular redefinition is frequently used because it is very unlikely that a user will key it accidentally, which is not the case of the prominent, and frequently pressed Escape key.

Time I made MY escape now, but I'll be back with more on error handling next month.

PCB DESIGNER

Geoff Bains, our electronics expert, has been putting Pineapple's new Printed Circuit Board designer under scrutiny.

Product: P.C.B.
Supplier: Pineapple Software
39 Brownlea Gds, Seven Kings,
Ilford, Essex IG3 9NL.
01-599 1476
Price: £97.75 inc. VAT

Anyone who has spent time up to their elbows in ferric chloride in the kitchen sink knows the importance of getting the design of a printed circuit board right before production.

Pineapple's PCB is an ingenious ROM package which makes the whole business of making your own printed circuit boards much more professional.

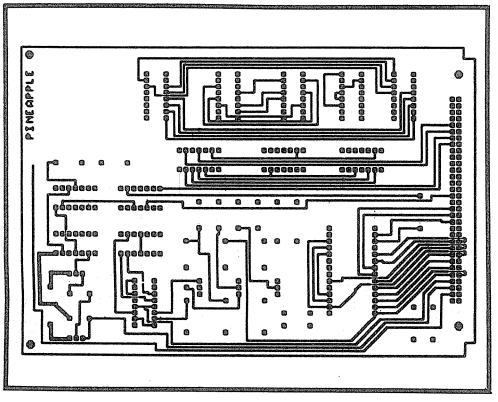
PCB will not help you to actually route the tracks on a circuit board design. There are software packages to do this but they are outside the realms of the humble Beeb. Instead, PCB is a highly specialised drawing package to bring professional PCB designs within any Beeb owner's reach.

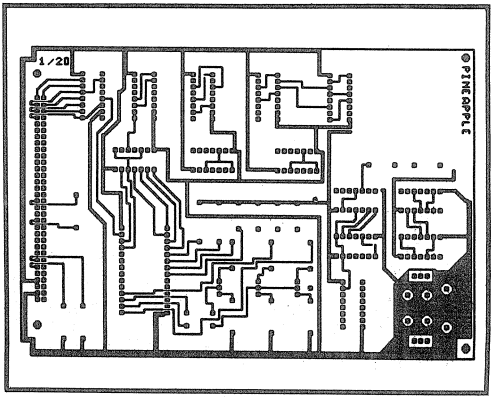
The biggest problem of producing a PCB design is getting all the information in front of you at one time while still leaving it readable. A complex design can consist of arrangements of tracks on both sides of the board, the component layout and even the component markings for a silk screen printing if the board is a professional production model. All this makes pencil and paper a messy and unreliable solution.

PCB allows any or all of these separate parts of a design to be displayed and edited on the screen at the same time. The individual 'layers' of the PCB design can then be printed out as finished artwork for the production process.

The program has two screens, each very similar to the other. The first screen is used to create the board shape, component layout and labelling. The software can cope with any board shape of up to 8 in by 5.6 in. The board outline is displayed on a mode 1 screen in yellow with Ø.1 in graduations around the edge. This is the smallest spacing for components and tracks.

Along the base of the screen are the symbols to place on the board. There are three sizes of 'roundels' - the circles of copper with a hole in the centre for soldering in a component lead - resistors, integrated circuits and lines (the last three in one of two orientations). These can be selected either with a cross-hair cursor under the control of the cursor keys or with a function key which cycles through them.





Once a symbol is chosen it can be placed on the board at the cursor position by pressing Return. At this point the size of the component can be changed with Shift and the cursor keys. When the length of the integrated circuit symbols is changed more or less connection pins are provided as well as the actual dimensions altered.

Text can also be added. This can be in any of four directions and in two sizes - the 'normal' mode 1 size and a small size for component labelling. Once a component or other marking has been placed on the board it can be picked up again and moved, changed in size or deleted. In this way the component layout and labelling is built up on the screen.

The next stage is to route the tracks between all the connections. This is done on the second screen. The process of routing the tracks on one or both sides of the board is similar to placing the components. One of three track widths is chosen from the menu at the base of the screen and a track is painted between roundels. The program automatically connects the tracks to a roundel if, and only if, the cursor is placed exactly over the connection point. In this way the tracks can be routed just where they are needed.

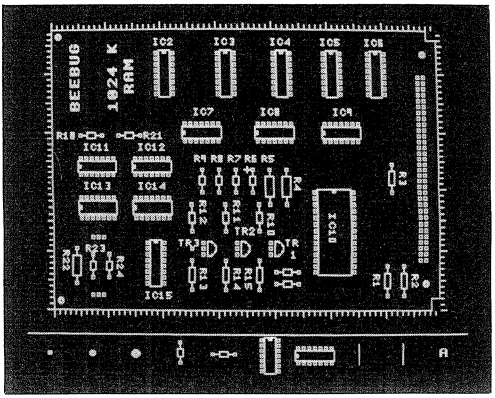
Again, text can be added in four directions, to appear on the final board as copper lettering. Tracks (or letters) can be deleted by filling them in with black. Alternatively a large area of the board, bounded by tracks, can be filled in to appear as an area of copper on the final board.

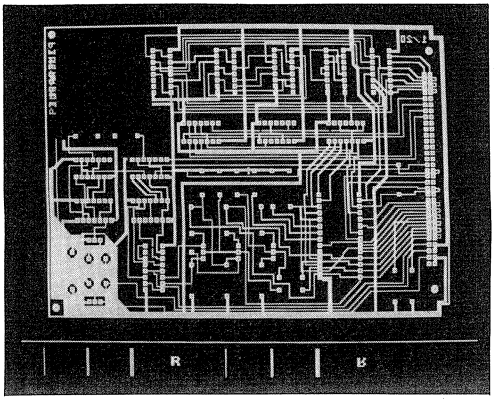
There are two colours of tracks that can be placed on the board - red and blue. These correspond to the two sides of a double sided circuit board. With PCB both sides of the board can be seen at the same time but are still easily recognised as separate, though use on a monochrome screen might make the red 'side' difficult to see.

Once the board has been fully drawn out, it can be stored on disc or printed out. The printout can be either the same size as the finished board or twice as large. PCB requires an Epson-compatible printer with the quadruple density graphics mode. This produces a very dense printout with a greater resolution than the board displayed on the screen. The roundels appear rounded on the printout, but decidedly pixelated on the screen. Tracks routed between two adjacent integrated circuit connection roundels appear to touch the roundels on the screen but on paper a special stretch of thin track is automatically printed. The printout can then be used directly to produce very adequate PCBs using one of the proprietary methods for this.

PCB does nothing you cannot do with drafting pen, or Letraset, and paper (and a good deal of time). However, this software does make the whole process of circuit board design considerably easier. It is easy to learn and generally well produced. This kind of package will never find mass appeal but for the limited number of Beeb users interested in DIY electronics, Pineapple has provided a most useful product.

B





Interactive 3D Updated

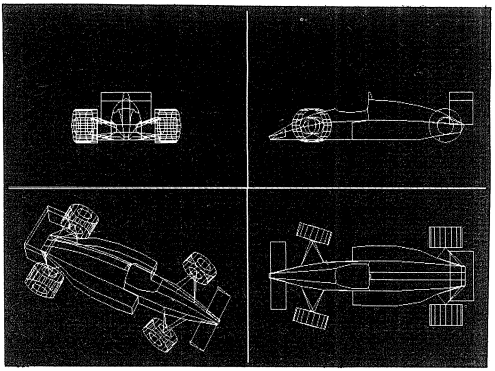
Design Dynamics have released a new version of their successful CAD package, Interactive 3D. Mike Williams reports on the new features.

Product: Interactive 3D (update)
Supplier: Design Dynamics
8 Meadow Way, Ampthill,
Bedford MK45 2QX.
(0525) 402447
Price: £12.95 (disc) £8.95 (tape)

Design Dynamics' Interactive 3D was first reviewed in BEEBUG in Vol.4 No.10 (April 1986), and was well recommended. For a detailed description of the package, readers are referred back to that original review. Now, a new version has been released, with a number of additional features and other improvements, but still selling at the same price as the original.

The package remains essentially as before, with a display showing perspective and First Angle Projection views. Drawing takes place on the perspective view, using six function keys to control movement in the X, Y and Z directions. There is also a separate cursor which can be moved two-dimensionally over the drawing to locate particular points as required. As drawing proceeds, the results are shown in all views simultaneously, and values of the current X, Y and Z co-ordinates are displayed at the top of the screen. While this does take some getting used to, it does all work very well. It is also very easy to return to points on the drawing and re-align these, and any lines connected to that point are automatically repositioned at the same time.

One of the principal new features is an automatic circle (and cylinder) drawing function. Circles are drawn about the 'previous' point and in a plane perpendicular to the axis joining the 'current' and 'previous' points. Cylinders are similarly drawn about an axis joining



those two points. In both cases, the radius is specified by the user. Both features work well and speedily, and form a useful addition to the package.

Design Dynamics also claims that accuracy has been improved, particularly where the zoom facility is concerned. Complete accuracy is now maintained within the eight-times range of this feature, unlike the previous version which would not always return precisely to the original co-ordinates after using zoom.

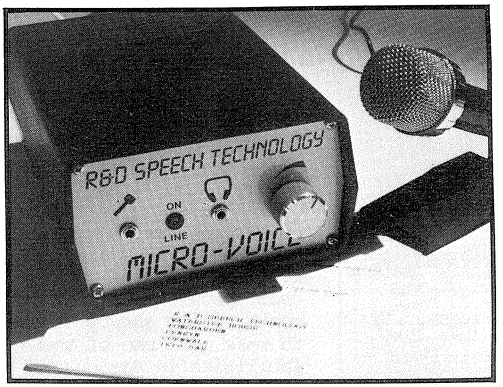
The other aspect which has been substantially improved is the production of hard copy. Previously a screen dump to Epson-compatible printers was provided, but this has proved too limiting. Now screens can be saved to disc at any time, and subsequently reloaded and dumped to a printer. This allows any suitable printer dump software to be used. It also means that screens produced by Interactive 3D can be enhanced using other software such as the BEEBUGSOFT Design package.

For its price, and within the memory limitations of the model B, Interactive 3D represents excellent value for money for all those interested in three-dimensional drawings. The package has already proved popular with schools, and this new version can only enhance Design Dynamics reputation in this field. BB

Note: Existing users of Interactive 3D can upgrade to the new version. Send the old disc with £3 to Design Dynamics. The upgrade includes manual supplement, keystrip and conversion program.

Speech Recognition

At approximately £17 a word, this speech recognition unit from R & D Speech Technology is hardly cheap. Mike Williams and Alan Webster have been talking to their Beebs to compile this report.



Product: Micro Voice
Supplier: R & D Speech
Waterside House, Ponsharden,
Penryn, Cornwall TR10 8AR.
(0326) 75290
Price: £171.35 inc. VAT

R & D Speech Technology have come up with the first speech recognition system for the BBC micro, and a very good effort it is indeed.

Housed in an attractive black box, the unit plugs into the micro via the 1MHz bus, and power is taken from the Beeb's own power supply. Any other device requiring power from the computer can then be plugged into the extension power socket at the back of the speech box.

The unit is supplied with a rather sparse booklet, a microphone, a demo disc or cassette, and software on ROM. There are no ROM fitting instructions, although R & D will fit the ROM for a small charge (if you send your machine to them).

The unit basically compares words or phrases spoken into the microphone with 'templates' already stored in memory. These are created with the *TRAIN command, up to a maximum of ten, and all ten must be recorded before recognition can start. To filter out ambient noise, you are allowed to average the data by recording any phrase more than once.

Recognition is achieved by way of a *LISTEN command which will return the number of the template nearest to the input, and the degree of recognition. These are provided in two zero page locations (&70 and &71). The degree of recognition is measured by a number in the range 0 to 255. This information is intended to be used by your own programs.

The unit is also capable of sound synthesis, needing just a lead to connect it to your hi-fi or amplifier. The only drawback with the unit as it stands (and this must be a severe disadvantage) is that any phrase needing to be pronounced, must first be sent to R & D on audio cassette, to be transferred by them into a suitable data format.

This is an expensive way of doing things, and surprising considering you have already purchased equipment to convert sound into data. The resultant data can then be loaded back into memory (or blown as a PHrase ROM), and speech produced using *TALK.

There is a small demonstration supplied with the unit, and the quality of this was very clear. The unit can hold around 200 words in 16K, which is reasonable considering the quality of speech achieved.

The manual supplied barely covers operation of the unit, but does have a few small example programs for you to type in and try. The demonstration disc is nothing to get excited about either.

As present, this unit is little more than an expensive toy, even more so if the speech synthesis capability is to be used. The recognition and speech output are of excellent quality, but most home users, I suspect, would be more inclined to spend £170 on something a little more useful.

BEEBUG WORKSHOP OS Calls from Basic

If you dismissed OSWORD calls as being only for machine code programmers then take another look. Surac shows how they can be used just as easily in Basic.

Have you ever noticed that you can't use the "star" commands (e.g. *FX) from Basic quite as easily as you can other commands? And all the operating system calls such as OSWORD - are they really only for machine code programmers? This month I'll show you how to make more use of these facilities in Basic.

STAR COMMANDS

There are 2 limitations on star commands in Basic. Firstly, they must be at the end of a line, or the only command in it. Secondly, you can't incorporate variables into them. For instance, consider:

```
10000 IF <test> *FX16,1:
    A%=ADVAL(1)
```

and:

```
 1000 *FX8,baudrate
```

Both give "Bad Command".

The first case is unlikely but you might want to do it sometime. However, since you can't put both commands on the same line, they would have to go into a procedure or subroutine, controlled by the IF.

What makes the star commands act this way? To understand, you have to know a little about what is inside the computer. In particular, there are (normally) at least 2 sets of ROM-based software - Basic and the Operating System (OS). These are actually totally different programs, and while a Beeb MUST have an OS chip, the Basic is not essential.

The OS makes all the different bits of the computer work together. It reads the keyboard, drives the monitor, handles the cassette interface and does everything else connected with getting things into and out of the Beeb, and making them work together when they are inside. When you run Basic, or any other language, it uses the OS to do all these jobs for it.

You can give commands directly to the OS, without using machine code, by means of a star command. The "*" tells Basic (or any other language) that everything following it on that line is to go to the OS, and NOT to the language.

The OS takes the characters AFTER the star and handles them by its own rules. These specify the letters and numbers which are possible, such as the *FX commands described in the User Guide, and others such as *SAVE.

What happens when line 10000 is executed? The OS sees "FX16,1", and understands that alright, but then it comes across ":A%=" etc. As it does not understand this, it exits with an error. In the second case, "baudrate" is a Basic variable, but the OS does not know about Basic; after "*FX8," it expects a numeric character and without one it fails.

Those problems above are irritating, but we can get around them. Sometimes, though, we MUST get data from Basic into the OS. Perhaps a program has to save a block of memory, such as a screen display, to a file whose name has been input. You could try:

```
    12000 INPUT "Save as? "file$
    12010 *SAVE file$ 8000 +8000
```

But that won't work (why not?).

Help is at hand in the Command Line Interpreter (CLI), which passes a string direct to the OS. The CLI is briefly described in the User Guide, and you can get into it in two ways. With Basic II or later, the command OSCLI(string) does the job for you. In an older Beeb with Basic I, you need a special procedure PROCoscli. Before you first use this procedure, set up the buffer "clibuff" with a "DIM clibuff 30" instruction.

```
20000 DEF PROCoscli(str$)
20010 LOCAL X%,Y%
20020 $clibuff=str$
20030 X%=clibuff MOD 256
20040 Y%=clibuff DIV 256
20050 CALL &FFF7
20060 ENDPROC
```

To find which Basic you have, type REPORT<Return>. If the copyright message says 1981 you have Basic I, while 1982 means Basic II; anything later is even better. In the following examples, you should use "PROCoscli" in place of "OSCLI" if you have the early system.

Now we have a way of getting variable data into the OS from Basic - convert it into a string. Line 12010 could become: 12010 OSCLI("SAVE "+file$+" 8000 + 8000")
and the earlier examples are:

```
 10000 OSCLI("FX16,1"):A%=ADVAL(1)
 11000 OSCLI("FX8,"+STR$(baudrate))
```

Note that there is no "*" at the start of the sequence. The asterisk tells Basic that the rest of the line has to go to the OS, and OSCLI does that for you. The OS itself ignores the "*".

While there is not much point in using variables with most star commands - they can only have one sensible value - some can benefit from the OSCLI approach. *SAVE and *LOAD are obvious examples, but some of the *FX commands can be useful, too. For instance:

```
 13000 INPUT "Key repeat speed? "S%
 13010 OSCLI("FX12,"+STR$(S%))
```

OTHER OS CALLS

While the CLI is the normal way of getting into the OS from Basic, it's not the only one. The User Guide lists lots of OS routines which look very useful. Most of them are really only for machine code programmers and do things which Basic does already. Some, though, do extra things and it would be nice to be able to use them.

For example, you might need to know which real colour (screen colour) is linked with a given logical colour (i.e., the one in the COLOUR statement). There is an OSWORD call (see the User Guide again) to do the job. Since we can get into OSWORD with CALL &FFF1, we can write a function to do the job:

```
21000 DEF FNrealcol(logcol)
21010 LOCAL A%,X%,Y%
21020 ?colbuf=logcol
21030 X%=colbuf MOD 256
21040 Y%=colbuf DIV 256
21050 A%=&B
21060 CALL &FFF1
21070 =colbuf?1
```

There you have a machine code function from Basic, without needing to know assembler. Remember to reserve space for your buffer (e.g. 'DIM colbuf 4').

Another example. Suppose you want to read the pixel pattern of an existing character (see First Course Vol.5 No.6) so that you can, maybe after modifying the pixels, use VDU 23,... to set a reprogrammable character:

```
22000 DEF PROCreadchar(char)
22010 LOCAL A%,X%,Y%
22020 ?charbuf=char
22030 X%=charbuf MOD 256
22040 Y%=charbuf DIV 256
22050 A%=&A
22060 CALL &FFF1
22070 ENDPROC
```

The character bytes are now in the top 8 bytes of "charbuf" (previously defined with "DIM charbuf 8"). Set character 230 to an inverted version of this value with:

```
 14000 VDU 23,230
 14010 FOR I%=8 TO 1 STEP -1:
       VDU charbuf?I%:NEXT
```

To use the OS successfully in this sort of way, first read the User Guide to see what the call needs. If necessary, DIMension a buffer and then write a procedure or function which:

1. Saves A%, X% and Y% by making them LOCAL.
2. Sets the buffer, and points X% and Y% to it.
3. Sets A% to the right value.
4. CALLs the correct start address of the OS routine.

Those are only the skeletons of some ideas, but they show you how it isn't essential to know machine code in order to use some assembly programmer tricks. If you look at the User Guide carefully, you will find lots of similar ideas. Apart from OSCLI, they are all a bit specialised, but nevertheless very handy sometimes. BB

HINTS and tips HINTS and tips HINTS and tips HINTS and tips HINTS and tips

Shifting Case

All letters can be forced into upper or lower case by making use of the fact that ASCII codes have just one bit different between upper and lower case characters (bit 5). Lower case letters have a '1' and upper case letters a '0' in this position. So to compare a letter in upper case regardless of its case, use 'A% OR &20' in the comparison, where A% is the ASCII code of the entered letter. To convert to upper case use 'A% AND &DF'.

There is an OSBYTE routine to switch the keyboard to produce only upper or lower case letters and to achieve the same effect as pressing the Shift-Lock, Caps-Lock and the very useful Shift-Caps-Lock. This is the OSBYTE call with the accumulator set to 202, or *FX202. The following table gives the range of effects.

Value	Effect	Caps	Shift
48	Lower case	Off	Off
32	Upper Case	On	Off
16	Shift Lock	Off	On
64	Shift Lock	On	On
160	Shift-Caps Lock	On	Off

Mike Trace

Faster Beeb

Any program can be speeded up a little by switching off the analogue (joystick) port. The Beeb automatically looks at the analogue-to-digital converter many times a second to read values from it. Disabling this provides more time for your program. Use *FX16,0. Don't try this, though, if the program is using joysticks!

Derek Gray

Long Wordwise Plus Preview

Even the humble BBC model B can preview documents of any length in mode 3 or 0 (i.e. with 80 columns) by making use of a little-recognised aspect of Wordwise Plus. A long document should be split into manageable portions (say, 1000 words at a time) and saved as FILE1, FILE2 and so on. Each should have as its last line a call to the next in the chain as follows:

```
<f1>PF"FILEX"<f2>
```

where <f1> and <f2> are the green and white codes, respectively, produced with function keys 1 and 2 and X is the following file number.

With the first file in memory, selecting options 6 or 7 will preview or print the entire document, reading subsequent files from disc and carrying forward all headers, footers, page numbers, and so forth. At the end, the first file will still be in memory.

Denis Groombridge

More Memory for Disc Users

The 81 locations from &380 to &3D0 are reserved for workspace for the cassette filing system, and appear to be unused unless this filing system is called. This area makes an excellent additional user RAM scratch pad for the hard pressed disc user. It is ideal for passing parameter blocks in OSWORD calls and the like, and is even large enough to hold modest machine code routines, leaving the frequently used &A00 buffer area free for other purposes.

David Graham

Faster Analogue

The Beeb's analogue-to-digital converter is not very fast, taking 10ms to complete each conversion. It normally produces a 10 bit digital number from the voltage applied at the analogue port. For non-critical applications (such as joysticks) the speed can be increased by reducing the accuracy to only 8 bits.

Conversion then takes around 4ms. The accuracy is controlled by bit 3 of location &FEC0 (a register of the DAC chip). Resetting this bit gives 8 bit operation. The other bits control the likes of how many channels are used. To set the converter to use an 8 bit conversion type:

```
?&FEC0=?&FEC0 AND 247
```

For more information, refer to page 429 of the Advanced User Guide.

Derek Gray

POSTBAG

Fudging the Issue

I find the RAMSAFE program (BEEBUG Vol.4 No.9) very useful. As I have battery backed sideways RAM it remains in there quite happily and enables me to retain any other program I want, without any of the hassle involved in producing a ROM image. The only small inconvenience is that after *RLOAD I find I have to start listing a program before it will run.

D.P.Dyer

Derek Floyd, the author of RAMSAFE, confirms that a program *RLOADed from cold will only run as described. This is because TOP is not reset. The OLD command would correct this, but you cannot readily call a routine in one ROM (Basic) from within another.

There is a 'fudge' that appears to work, though it does waste up to 255 bytes of memory, and involves adding the following lines to the RAMSAFE program:

```
2812 LDA &73
2814 STA &13
```

The RAMSAFE program keeps a record of the last address transferred, but only to the nearest 256 byte boundary, and this can be used to reset TOP.

Inspiring Menu

Here is an alternative, simpler View file loader for the ADFS Menu program (Vol.5 Nos.4 & 5) - inspired by that of David Graham. All View files are saved to a directory VIEW, the function key buffer being used instead of an EXEC file. The modifications to the program are listed below:

```
2030 key$="*KEY0*WORD|MLOAD
 "+A$(F%-64)+"|M"
2040 OSCLIkey$
2050 *FX138 0 128
```

E. Mark Jolliffe

This alternative certainly has the merit of simplicity but requires that all your View files are kept in the one directory called VIEW. This does abandon one of the benefits of using the ADFS, the facility to organise one's files in different directories by function.

Live and Let Live

Would Peter Hill tell us why he wants 255 lives in Frak! (see Hints & Tips BEEBUG Vol.5 No.7)? I cannot see the enjoyment or sense of achievement if this alteration is made. I might use the tip to change to one life (one day when I get better) but I am sure Aardvark has set a sensible number of lives to make the game worth playing.

Brian Westbury

All we can say is that on the same day as we received Brian Westbury's letter, we received another, delighted with the information and seeking more.

Surprise, Surprise

I have found some interesting results using the printer check routine (see First Course, BEEBUG Vol.5 No.3). With the printer turned on and switched on-line, the routine responds correctly.

With the printer turned on but switched off-line, the routine gives the right response, and again if the printer is switched on-line and the routine re-run.

With the printer turned off initially, surprise number one; although two bytes have been sent to the printer, ADVAL(-4) shows 62 bytes free - only one of the bytes is still in the buffer. Surprise number two now occurs if the printer is switched on and the routine re-run. The routine responds as if the printer is still not connected. Surprise number three; if the routine is now run again it does produce the correct result.

Do you have any ideas that might explain the behaviour or solve the difficulties described?

Mike Markey

When the routine was first written, it sent only one test byte to the printer. It was because this proved unreliable that the routine was amended to send two bytes. This gave the correct results in tests, but is clearly not foolproof. I cannot explain where the missing bytes have gone, but it might be worth making the routine send four (or more) bytes and seeing if correct responses are then obtained in all cases.

GAMES REVIEWS By Daniel Gaster

Title: Repton 3
Supplier: Superior Software
Price: £9.95 (£11.95 on disc)
Rating: *****

Just when you thought that you had heard the last of Repton, Superior have released the next in the cycle. This time it's even bigger, more complicated, and with many extra features. The game still revolves around the little green reptile finding his way around a cavern (now packed with growing fungi, time bombs, time capsules and golden crowns, as well as skulls, rocks, spirits, keys, diamonds, monsters, etc.). A major improvement to the game is the addition of a screen and character designer that can be used to create a completely new-look game (just like BEEBUG's Pitfall Pete in Vol.5 No.7).

The size of the game is huge; there are 24 screens in all, and each has a password and must be completed in succession to finish the whole game. Everything about this game is good. The graphics are superlative, drawn in huge, colourful, well-animated sprites. The scrolling is quick, the sound is good and the game presents a huge challenge to those who prefer problem-solving to blasting 'aliens'. The puzzles are, in the main, fiendishly difficult; most of the screens are impossible to complete without lengthy practice. Highly recommended.

Title: Ravenskull
Supplier: Superior Software
Price: £9.95 (£11.95 on disc)
Rating: ****(*)

Ravenskull is another top-quality release from Superior, featuring a vast playing area. The idea of the game is to travel round a castle to recover a silver crucifix. Windowing is used (as in the Repton series) to move from room to room.

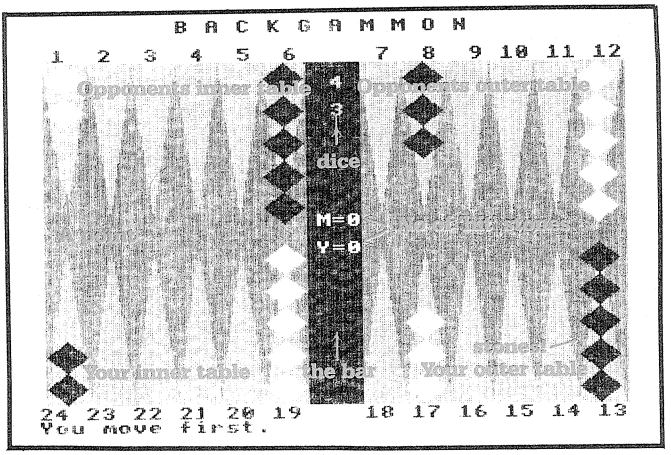
The game features are many and varied. There are over 25 different objects to find, use or avoid, including potions and scrolls, a bow and arrow, a cake, and some dynamite. You can chose which of four characters to be (warrior, elf, wizard or adventurer) and you can carry up to three objects at once.

The graphics are large and colourful and the title screen is about the best I have ever seen on the Beeb. The game is slightly slow to get into but the review copy allowed me into later sections of the game where the action hots up considerably. This game is a challenging graphic adventure, well worth the money.

BB

BACKGAMMON

As a rest from the more usual arcade-style of game, we present Michael Bell's excellent implementation of the up-market game of Backgammon. If you don't know how to play, now's your chance to learn.



Backgammon is a popular board game with something of an up-market image. In this version you play with the computer as your opponent, and the program prompts you for every move (and checks it as well). Even if you have not played Backgammon before, you will soon become quietly addicted.

THE RULES OF BACKGAMMON

The object of the game is quite simple: to move some 'stones' around a board, avoid being 'hit' by your opponent, and be the first to remove your stones from the playing area.

The board is split into four main sections (see figure 1), with the triangular points numbered from 1 to 24. Your opponent has to move the white stones anti-clockwise around the board, and you have to move the black stones clockwise.

The first objective is to move your 15 stones into your inner table (points 19-24), before you can begin to 'bear off' - remove the stones from the board.

Two dice are rolled to decide who goes first (the dice are shown at the top of the middle bar). When the dice have been 'thrown' you may move 1 or 2 stones by the relevant number of places shown on each dice. If a six and two are thrown, for instance, you may move one stone by six and then two, or two and then six, or you may move one stone by six and another by two. You cannot move more than one stone for each value of the dice. If a double is thrown, you may move four times the amount shown on one dice.

Each player may move a stone onto any unoccupied point, a point that has their own stones on it, or a point containing just one of their opponent's stones. Points occupied by more than one of your opponent's stones (a closed point) are out of bounds.

If you move to a point occupied by a single opposing stone, that stone becomes 'hit', and is moved to the centre bar. A player with one or more stones on the centre bar must move these back onto the board (using the dice to determine the start position) before moving any other piece. The number of stones on the bar is indicated by 'M' for your opponent, and 'Y' for yourself.

Once either player has moved all their stones into their respective inner table, they may then start to bear off. To bear off, a stone should be moved beyond the end of the board. If a four is thrown, any stone in positions one to four can be removed, or a stone moved by four points if it occupies positions five or six. The game ends with the first player to remove all fifteen of their stones.

If the loser has not removed any stones by the finish, the game is lost by a gammon (equivalent to 2 games). If

any of the loser's stones are still left in the winner's inner table, the winning margin is called a backgammon (equivalent to 3 games).

For further information on the game, and good playing strategies, refer to The Illustrated Backgammon Book by O. Jacoby and J. R. Crawford, published by Pan (ISBN 0 330 24234 2) at £3.50.

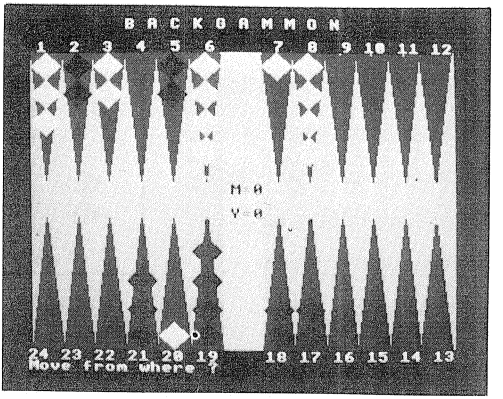
USING THE PROGRAM

Once typed in and safely saved, the program may be run to play the game. You will find prompts to help you select the appropriate move when it is your turn to play, and the program checks for any illegal moves. Do remember that if any of your stones has been hit, they must be returned to the playing area before any other moves take place. All you have to do now is get playing.

As the program is quite long and uses mode 1, users with PAGE set higher than &E00 (model B and B+ users with disc systems fitted) should add the following move-down routine to the start of the program.

```
 0 *TAPE
 1 *K.0 FORA%=0TO(TOP-PAGE)STEP4:A%!&E00=
   A%!PAGE:NEXT|MPAGE=&E00|MOLD|MGOTO10|M
 2 *FX138 0 128
 3 END
```

```
   10 REM Program Backgammon
   20 REM Version B0.2
   30 REM Author  Michael Bell
   40 REM Beebug  Jan/Feb 1987
   50 REM Program subject to copyright
   60 :
  100 ONERRORMODE7:REPORT:PRINT" at line
 ";ERL:END
  110 MODE1:VDU23,1,0;0;0;0;:DIMB$(24)
  120 REPEAT
  130 PROCscreen:PROCsetup:PROCpieces
  140 REPEAT:PROCdice:UNTILD1%<>D2%
  150 IFD1%>D2% PRINTTAB(1,30)"You move
first."ELSEPRINTTAB(1,30)"I move first."
  160 PROCpause:PROCspace
  170 IFD1%>D2% PROCplayer
  180 REPEAT
  190 PROCcomputer
  200 IFCB%=15THEN220
  210 PROCplayer
  220 UNTILPB%=15OR CB%=15
  230 IFCB%=15 PRINTTAB(1,30)"I win by "
;FNscore("P",0);P$
  240 IFPB%=15 PRINTTAB(1,30)"You win by
 ";FNscore("C",18);P$
  250 PROCpause:PROCspace
  260 INPUTTAB(1,30)"Another game ",A$
  270 UNTILA$="N"OR A$="n"
```




```
  280 MODE7:PRINTTAB(1,1)"Bye"
  290 END
  300 :
 1000 DEFPROCstone(X%,Y%,C%)
 1010 GCOL0,C%
 1020 MOVEX%,Y%:MOVEX%-40,Y%+36
 1030 PLOT85,X%+40,Y%+36:PLOT85,X%,Y%+72
 1040 ENDPROC
 1050 :
 1060 DEFPROCscreen
 1070 VDU19,2,2,0,0,0
 1080 L1%=100:L2%=450:CLS
 1090 GCOL0,2
 1100 MOVE50,100:MOVE1250,100
 1110 PLOT85,50,900:PLOT85,1250,900
 1120 GCOL0,1
 1130 MOVE50,L1%:MOVE95,L2%:PLOT85,140,L
1%
 1140 FORX%=140TO500STEP90
 1150 MOVEX%+45,L2%:PLOT85,X%+90,L1%
 1160 NEXT
 1170 MOVE700,L1%:MOVE745,L2%:PLOT85,790
,L1%
 1180 FORX%=790TO1150STEP90
 1190 MOVEX%+45,L2%:PLOT85,X%+90,L1%
 1200 NEXT
 1210 IFL1%=100 L1%=900:L2%=550:GOTO1130
 1220 GCOL0,3
 1230 MOVE590,100:MOVE700,100
 1240 PLOT85,590,900:PLOT85,700,900
 1250 PRINTTAB(2,3)"1  2  3  4  5  6
 7  8  9 10 11 12"
 1260 PRINTTAB(1,29)"24 23 22 21 20 19
  18 17 16 15 14 13"
```

```
 1270 PRINTTAB(10,1)"B A C K G A M M O N
"
 1280 ENDPROC
 1290 :
 1300 DEFPROCsetup
 1310 FORL1%=0TO24:B$(L1%)="000":NEXT
 1320 B$(1)="P2":B$(12)="P5":B$(17)="P3"
:B$(19)="P5"
 1330 B$(6)="C5":B$(8)="C3":B$(13)="C5":
B$(24)="C2"
 1340 PO%=0:CO%=0:PH%=5:CH%=5
 1350 PP$="P0P1P2P3P4P5P6P7P8P9C100":PB%
=0
 1360 CC$="C0C1C2C3C4C5C6C7C8C9P100":CB%
=0
 1370 ENDPROC
 1380 :
 1390 DEFPROCpieces
 1400 PROCupdate(1,0):PROCupdate(12,0)
 1410 PROCupdate(17,0):PROCupdate(19,0)
 1420 PROCupdate(6,3):PROCupdate(8,3)
 1430 PROCupdate(13,3):PROCupdate(24,3)
 1440 ENDPROC
 1450 :
 1460 DEFPROCdice
 1470 COLOUR1:COLOUR131
 1480 DI%=RND(-TIME)
 1490 FORL1%=1TO10
 1500 D1%=RND(6):D2%=RND(6)
 1510 FORL2%=1TO900:NEXT
 1520 PRINTTAB(20,5);D1%:PRINTTAB(20,7);
D2%
 1530 NEXT
 1540 COLOUR3:COLOUR128:VDU 7
 1550 ENDPROC
 1560 :
 1570 DEFPROCplayer
 1580 P%=1:M%=2:T%=1
 1590 PROCdice
 1600 IFFNcanpmove PROCsorry:ENDPROC
 1610 IFD1%=D2% M%=4
 1620 REPEAT
 1630 IFNOTFNofboard AND L1%=1 THEN1650
 1640 IFNOTFNofboard THEN1700ELSE1940
 1650 REPEAT
 1660 INPUTTAB(1,30)"Move from where ",F
%
 1670 IFNOTFNcheck1(F%) VDU7
 1680 PROCspace
 1690 UNTILFNcheck1(F%)
 1700 IFD1%<>D2% PROCsingle
 1710 IFT%=0 AND PH%<>15 THEN1630
 1720 IFD1%<>D2% IFT%=0AND NOT(F%+D1%>24
 OR F%+D2%>24) VDU7:GOTO1630
 1730 IFT%=0 IFD1%>D2% D1%=-100
 1740 IFT%=0AND D1%>0AND D1%<>D2% D2%=-1
00
 1750 IFD1%=D2% AND D1%>0 PROCdouble
 1760 IFT%=0AND PH%<>15OR P%=0 THEN1630
 1770 INPUTTAB(1,30)"Are you sure(Y/N) "
,A$
 1780 PROCspace
 1790 IFA$="N"OR A$="n" THEN1630
 1800 IFB$(T%)="C1"AND T%<7 CH%=CH%-1
 1810 IFB$(T%)="C1" CO%=CO%+1:B$(T%)="00
0"
 1820 IFPO%>0 PO%=PO%-P%
 1830 B$(F%)="P"+STR$(VAL(MID$(B$(F%),2)
)-P%)
 1840 IFVAL(MID$(B$(F%),2))=0 B$(F%)="00
0"
 1850 B$(T%)="P"+STR$(VAL(MID$(B$(T%),2)
)+P%)
 1860 IFT%>18AND F%<19 PH%=PH%+P%
 1870 IFT%=0 PB%=PB%+P%
 1880 PROCupdate(F%,0):PROCupdate(T%,0)
 1890 M%=M%-P%
 1900 IFD1%<>D2%AND T%-F%=D1% D1%=-100
 1910 IFD1%<>D2%AND T%-F%=D2% D2%=-100
 1920 IFFNcanpmove PROCsorry:M%=0
 1930 IFPB%=15 M%=0
 1940 UNTILM%=0
 1950 ENDPROC
 1960 :
 1970 DEFFNcheck1(A%)
 1980 IFA%>24OR A%<1 =FALSE
 1990 IFLEFT$(B$(A%),1)<>"P" =FALSE
 2000 =TRUE
 2010 :
 2020 DEFFNcheck2(A%)
 2030 IFA%>M%OR A%<0 =TRUE
 2040 IFVAL(MID$(B$(F%),2))<A% =TRUE
 2050 =FALSE
 2060 :
 2070 DEFFNcheck3(A%)
 2080 IFA%>24OR A%<1 =TRUE
 2090 IFLEFT$(B$(A%),1)="C" AND MID$(B$(
A%),2,1)<>"1" =TRUE
 2100 IFA%<=F% =TRUE
 2110 IFA%-F%<>D1%AND A%-F%<>D2% =TRUE
 2120 =FALSE
 2130 :
 2140 DEFPROCupdate(A%,C%)
 2150 IFA%=0 ENDPROC
 2160 COLOUR0:COLOUR131
 2170 PRINTTAB(19,15)"M=";CO%
 2180 PRINTTAB(19,17)"Y=";PO%
 2190 COLOUR3:COLOUR128
 2200 IFA%<7 X%=90*A%+5
 2210 IFA%>6 AND A%<13 X%=90*A%+115
 2220 IFA%>12 AND A%<19 X%=90*(25-A%)+11
5
 2230 IFA%>18AND A%<25 X%=90*(25-A%)+5
 2240 IFA%<13 L1%=890:L2%=550:L3%=-50 EL
SEL1%=100:L2%=450:L3%=50
 2250 GCOL0,2
 2260 MOVEX%,L2%:DRAWX%,L2%+L3%
 2270 MOVEX%-45,L1%:MOVEX%-45,L2%+L3%:PL
OT85,X%,L2%+L3%
```

```
2280 MOVEX%+45,L2%+L3%:PLOT85,X%+45,L1%
2290 GCOL0,1
2300 MOVEX%-45,L1%:PLOT85,X%,L2%
2310 L3%=VAL(MID$(B$(A%),2))
2320 IFL3%=0 ENDPROC
2330 IFA%>12 THEN2380
2340 IFL3%<5 L1%=895-L3%*75 ELSEL1%=520
2350 FORL2%=L1%TO820STEP75
2360 PROCstone(X%,L2%,C%)
2370 NEXT
2380 IFA%<13 ENDPROC
2390 IFL3%<5 L1%=L3%*75+25 ELSEL1%=400
2400 FORL2%=100TOL1%STEP75
2410 PROCstone(X%,L2%,C%)
2420 NEXT
2430 ENDPROC
2440 :
2450 DEFPROCdouble
2460 REPEAT
2470 L1%=0
2480 INPUTTAB(1,30)"How many pieces(0 t
o Abort) ",P%
2490 IFFNcheck2(P%) VDU7ELSE L1%=1
2500 IFP%=0 THEN2540
2510 T%=F%+D1%
2520 IFT%>24AND PH%<15 VDU7:L1%=0
2530 IFT%<25 IFFNcheck3(T%) VDU7:L1%=0
2540 PROCspace
2550 UNTILL1%=1
2560 IFPH%=15 AND T%>24 T%=0
2570 ENDPROC
2580 :
2590 DEFPROCsingle
2600 REPEAT
2610 INPUTTAB(1,30)"Move to(0 to abort/
Bear off) ",T%
2620 IFT%=0 THEN2640
2630 IFFNcheck3(T%) VDU7:T%=-1
2640 PROCspace
2650 UNTILNOTFNcheck3(T%)OR T%=0
2660 ENDPROC
2670 :
2680 DEFPROCspace
2690 PRINTTAB(1,30)SPC38
2700 ENDPROC
2710 :
2720 DEFFNofboard
2730 L1%=0:B$(0)="P"+STR$(PO%):A$="ZZ"
2740 IFPO%=0 L1%=1:=FALSE
2750 IFD1%>0 A$=LEFT$(B$(D1%),2)
2760 IFINSTR(PP$,A$)<>0 F%=0:=FALSE
2770 IFD2%>0 A$=LEFT$(B$(D2%),2)
2780 IFINSTR(PP$,A$)<>0 F%=0:=FALSE
2790 PROCsorry
2800 M%=0:=TRUE
2810 :
2820 DEFFNcanpmove
2830 IFD1%=-100AND D2%=-100OR M%=0 =FAL
SE
2840 IFPH%=15 =FALSE
2850 L1%=0
2860 IFD1%>D2% L3%=D1% ELSE L3%=D2%
2870 FORL2%=0TO24-L3%
2880 IFD1%>0 IFLEFT$(B$(L2%),1)="P" IFI
NSTR(PP$,LEFT$(B$(L2%+D1%),2))<>0 L1%=1
2890 IFD2%>0 IFLEFT$(B$(L2%),1)="P" IFI
NSTR(PP$,LEFT$(B$(L2%+D2%),2))<>0 L1%=1
2900 NEXT
2910 IFL1%=0 =TRUE ELSE=FALSE
2920 :
2930 DEFPROCsorry
2940 PRINTTAB(1,30)"Sorry you cannot mo
ve."
2950 PROCpause:PROCspace
2960 ENDPROC
2970 :
2980 DEFPROCnomoves
2990 PRINTTAB(1,30)"I cannot move";SPC2
4
3000 PROCpause:PROCspace
3010 ENDPROC
3020 :
3030 DEFPROCown(A$)
3040 FORL1%=1TO24
3050 IFD1%>0 L2%=L1%+D1% ELSE L2%=25
3060 IFD2%>0 L3%=L1%+D2% ELSE L3%=25
3070 IFL1%-ABS(D1%)>0 IFB$(L1%)="C1"AND
 LEFT$(B$(L1%-D1%),1)="C" F%=L1%:T%=L1%-
D1%
3080 IFL1%-ABS(D2%)>0 IFB$(L1%)="C1"AND
 LEFT$(B$(L1%-D2%),1)="C" F%=L1%:T%=L1%-
D2%
3090 IFL2%<25 IFB$(L1%)=A$AND LEFT$(B$(
L2%),1)="C" F%=L2%:T%=L1%
3100 IFL3%<25 IFB$(L1%)=A$AND LEFT$(B$(
L3%),1)="C" F%=L3%:T%=L1%
3110 NEXT
3120 ENDPROC
3130 :
3140 DEFPROCbearoff
3150 IFCH%<>15 ENDPROC
3160 IFD1%>D2% L1%=D1%:D1%=-100
3170 IFL1%=0AND D2%>D1% L1%=D2%:D2%=-10
0
3180 IFD1%>0 IFD1%=D2% L1%=D1%
3190 IFLEFT$(B$(L1%),1)="C" F%=L1%:T%=0
:ENDPROC
3200 FORL2%=L1%TO6
3210 IFLEFT$(B$(L2%),1)="C" IFINSTR(CC$
,LEFT$(B$(L2%-L1%),2))<>0 F%=L2%:T%=L2%-
L1%
3220 NEXT
3230 IFF%>0 ENDPROC
3240 FORL2%=1TOL1%-1
```

```
 3250 IFLEFT$(B$(L2%),1)="C" F%=L2%:T%=0
 3260 NEXT
 3270 ENDPROC
 3280 :
 3290 DEFPROCblock
 3300 L2%=0
 3310 FORL1%=0TO24
 3320 IFLEFT$(B$(L1%),1)="C" L2%=L2%+1
 3330 IFLEFT$(B$(L1%),1)="P"AND L2%<>15
L2%=-1
 3340 NEXT
 3350 IFL2%=-1 ENDPROC
 3360 IFD1%<=D2% L1%=D2% ELSE L1%=D1%
 3370 FORL2%=24-L1%TO1STEP-1
 3380 IFB$(L2%)="000" IFLEFT$(B$(L2%+D1%
),1)="C" IFLEFT$(B$(L2%+D2%),1)="C" F%=L
2%+D1%:F2%=(L2%+D2%)*-1:T%=L2%
 3390 IFD1%=D2%AND F2%<>0AND VAL(MID$(B$
(L2%+D2%),2,1))<2 F%=0:F2%=0
 3400 NEXT
 3410 ENDPROC
 3420 :
 3430 DEFPROCcomputer
 3440 PRINTTAB(1,30)"My move!!!"
 3450 M%=2:F2%=0:F%=0
 3460 PROCdice
 3470 IFD1%=D2% M%=4
 3480 REPEAT
 3490 L1%=0:PROCbearoff
 3500 IFCO%>0 PROCoff
 3510 IFM%=1AND D1%=D2% D2%=-100
 3520 IFF%=25 THEN3730
 3530 IFF%>25 THENF%=0:GOTO3570
 3540 IFF%<>0 THEN3570
 3550 PROCbestmove
 3560 IFF%=0 M%=0:GOTO3730
 3570 PRINTTAB(1,30)"I move from ";F%;"
to ";T%;SPC10
 3580 PROCpause
 3590 IFB$(T%)="P1"AND T%>18 PH%=PH%-1
 3600 IFB$(T%)="P1"AND T%>0 PO%=PO%+1:B$
(T%)="000"
 3610 B$(F%)="C"+STR$(VAL(MID$(B$(F%),2)
)-1)
 3620 IFVAL(MID$(B$(F%),2))=0 B$(F%)="00
0"
 3630 B$(T%)="C"+STR$(VAL(MID$(B$(T%),2)
)+1)
 3640 IFCO%>0 CO%=CO%-1
 3650 IFT%<7AND F%>6 CH%=CH%+1
 3660 PROCupdate(F%,3):PROCupdate(T%,3)
 3670 IFT%=0 CB%=CB%+1
 3680 M%=M%-1
 3690 IFD1%<>D2%AND F%-T%=D1% D1%=-100
 3700 IFD1%<>D2%AND F%-T%=D2% D2%=-100
 3710 F%=0
 3720 IF CB%=15 M%=0
 3730 UNTILM%=0
 3740 PROCspace
 3750 ENDPROC
 3760 :
 3770 DEFPROCoff
 3780 IFD1%>0 A$=LEFT$(B$(25-D1%),2) ELS
E A$="ZZ"
 3790 IFINSTR(CC$,A$)<>0 F%=26:T%=25-D1%
:D1%=D2%:ENDPROC
 3800 IFD2%>0 A$=LEFT$(B$(25-D2%),2) ELS
E A$="ZZ"
 3810 IFINSTR(CC$,A$)<>0 F%=26:T%=25-D2%
:D2%=D1%:ENDPROC
 3820 M%=0:F%=25
 3830 PROCnomoves
 3840 ENDPROC
 3850 :
 3860 DEFPROCbestmove
 3870 IFF2%<0 F%=ABS(F2%):F2%=0:ENDPROC
 3880 F%=0:T%=0
 3890 IFPH%=15THEN 4000
 3900 PROCown("C1")
 3910 FORL1%=0TOT%
 3920 IFLEFT$(B$(L1%),1)="P" L1%=T% ELSE
 L2%=L1%
 3930 NEXT
 3940 IFT%>0AND L2%=T% F%=0
 3950 IFF%<>0 ENDPROC
 3960 IFD1%>0AND D2%>0 PROCblock
 3970 IFF%<>0 ENDPROC
 3980 PROCown("P1")
 3990 IFF%<>0 ENDPROC
 4000 IFD1%<=D2% L1%=D2% ELSE L1%=D1%
 4010 FORL2%=1TO24-L1%
 4020 IFD1%>0 IFLEFT$(B$(L2%),1)<>"P"AND
 LEFT$(B$(L2%+D1%),1)="C" F%=L2%+D1%:T%=
L2%
 4030 IFD2%>0 IFLEFT$(B$(L2%),1)<>"P"AND
 LEFT$(B$(L2%+D2%),1)="C" F%=L2%+D2%:T%=
L2%
 4040 NEXT
 4050 IFF%<>0 ENDPROC
 4060 PROCnomoves
 4070 ENDPROC
 4080 :
 4090 DEFPROCpause
 4100 FORL1%=1TO20000:NEXT
 4110 ENDPROC
 4120 :
 4130 DEFFNscore(A$,A%)
 4140 L2%=0:P$=" games."
 4150 FORL1%=1+A%TO 6+A%
 4160 IFLEFT$(B$(L1%),1)=A$ L2%=1
 4170 NEXT
 4180 IFL2%=1OR PO%>0OR CO%>0 =3
 4190 IFPB%=0OR CB%=0 =2
 4200 P$=" game.":=1
```

BEEBUG MAGAZINE is produced by BEEBUG Ltd.
Editor: Mike Williams
Technical Assistant: Alan Webster
Production Assistant: Yolanda Turuelo
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft
Additional thanks are due to Sheridan Williams, Adrian Calcraft, Geoff Bains, John Yale and Tim Powys-Lybbe.

Editorial Address
BEEBUG
Dolphin Place
Holywell Hill
St. Albans
Herts. AL1 1EX

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 £40 per page, but please give us warning of anything substantial that you intend to write. A leaflet, 'Notes of Guidance for Contributors', is available on receipt of an A5 (or larger) SAE.

In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "View", or other means, but please ensure an adequate written description of your contribution is also included. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month, plus one of £15 for a hint or tip deemed to be exceptionally good.

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

BEEBUG MEMBERSHIP

Send all applications for membership, membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be pounds sterling drawn (for cheques) on a UK bank.

MEMBERSHIP SUBSCRIPTION RATES

£ 6.90 6 months (5 issues) UK ONLY
£12.90 - 1 year (10 issues) UK, B.F.P.O., Channel Islands
£19.00 - Rest of Europe £23.50 Middle East
£26.00 Americas & Africa. £28.00 Elsewhere

BACK ISSUES
(Members only)

Volume	Single issues	Volume sets (10 issues)
1	40p	£2.50 (8 issues) – issues 1 and 8 out of stock
2	50p	£3.50
3	70p	£5.50
4	£1	£8.50
5	£1.30	–

Please add the cost of post and packing as shown:

DESTINATION	First issue	Each subsequent issue
UK, BFPO + CH. IS.	40p	20p
Europe + Eire	75p	45p
Elsewhere	£2	85p

All overseas items are sent airmail (please send a sterling cheque). 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 that require an invoice. Note that there is no VAT on magazines.

Back issues are for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

BEEBUG
Dolphin Place
Holywell Hill
St. Albans
Herts. AL1 1EX

Hotline for All non-technical queries
i.e. orders, membership subscription, membership queries etc.

St. Albans (0727) 40303
Manned Mon-Fri 9am-4.30pm

(24hr Answerphone Service for Access/ Visa orders and subscriptions

Technical Queries
St. Albans (0727) 60263
Mon-Fri 10am-4pm

If you require members discount it is essential to quote your membership number and claim the discount when ordering.

Produced by Head Office Design (0782) 286375 ISSN – 0263 – 7561

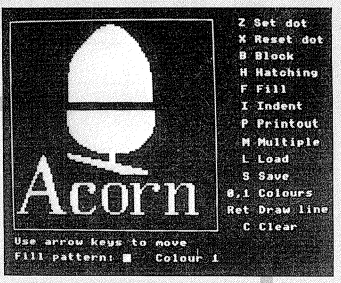
Magazine Cassette/Disc

Jan/Feb 1987
CASSETTE/DISC CONTENTS

SIDEWAYS RAM BUFFERS — useful utility for squeezing the last drop of memory space from your machine.
PERSONALISED LETTER HEADINGS — most useful logo designer and conversion program for producing your own fancy letterheads.
THE MASTER SERIES
TRANSFERRING FILES BETWEEN DFS AND ADF — comprehensive set of function key definitions.
EXEC FILES UPDATED — more useful EXEC files for Master users.
MAINTAINING STATUS — useful back-up facility for your Master's configuration.
BUSINESS GRAPHICS — complete working program including latest 3D and pie chart options, plus a sample data file.
SORTING DATA FILES — complete sort program, repeated from last month, to modify for use with your own data files.
VIRTUAL ARRAYS — full driver program plus complete demonstration of this useful technique.
BEEBUG WORKSHOP — set of examples OS routines called from Basic.
BACKGAMMON — good implementation of this up-market board game for beginner and expert alike.

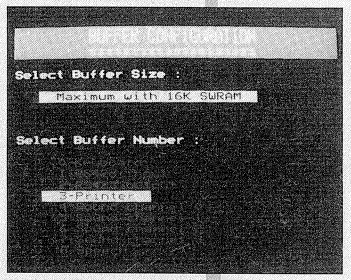
EXTRA FEATURES THIS MONTH

MAGSCAN — data for this issue of BEEBUG (Vol. 5 No. 8).

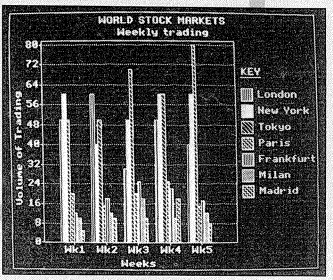


Letter Headings

Sideways RAM Buffers



Business Graphics



All this for £3.00 (cass) £4.75 (disc) +50p p&p.
Back issues (disc since Vol. 3 No. 1, cass since Vol. 1 No. 10) available at the same prices.

Subscription rates	DISC UK	CASS UK	DISC O'seas	CASS O'seas
6 months (5 issues)	£25.50	£17	£30	£20
12 months (10 issues)	£50	£33	£56	£39

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

Cassette subscriptions can be commuted to disc subscription on receipt of £1.70 per issue of the subscription left to run.

All subscription and individual orders to
BEEBUG, Dolphin Place, Holywell Hill, St. Albans, AL1 1EX.

DYNAMIC DISCS

FROM BEEBUG

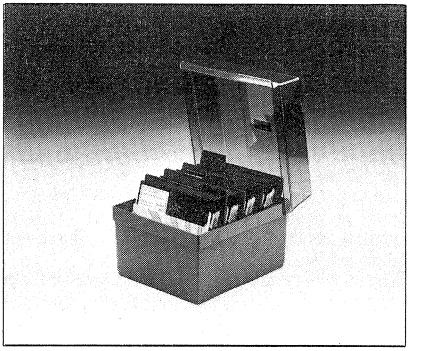
Our range of top quality 5¼" blank discs for the BBC Micro and Master 128 are now available at unbeatable prices.

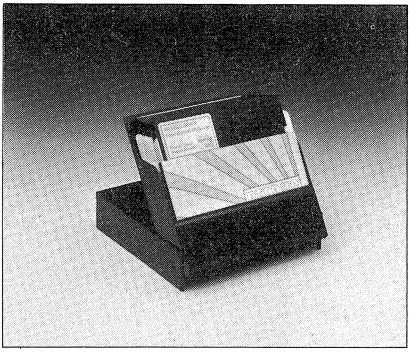
These discs are from the UK's leading disc manufacturer, individually tested and guaranteed for life. They are specially produced and boxed for Beebug. Don't settle for anything less than the best.

Beebug members may claim a 5% discount on the price. If you are not yet a member please phone for an application form.

48 TPI DOUBLE DENSITY

		PRICE	MEMBERS
10	S/S	£9.90	£9.40
25	S/S	£23.00	£21.85
50	S/S	£48.00	£45.60
10	D/S	£10.42	£9.90
25	D/S	£26.47	£25.15
50	D/S	£50.00	£47.50





96 TPI QUAD DENSITY

		PRICE	MEMBERS
10	S/S	£10.42	£9.90
25	S/S	£26.47	£25.15
50	S/S	£50.00	£47.50
10	D/S	£10.90	£10.35
25	D/S	£29.00	£27.55
50	D/S	£53.00	£50.30

- **Prices include FREE Storage Box**
- **Official Orders welcome**
- **ALL PRICES INCLUDE V.A.T.**
- UK Postage 10's — £1.00; 25's & 50's — £3.75

5% MEMBERS DISCOUNT

ORDER FORM

Please send me_____Qty of _____ @ £_____ (Plus Postage £_____)

I enclose a cheque for £_____
or Please debit my Access/Visa No. | | | | |

Name: _____ **Membership No:** _____

Address: _____

_____

Post Code: _____

NOTE: To claim members prices it is essential to quote membership no. If not yet a member add **£12.90** membership fee, to claim members discount.

Send to: Beebug Mail Order, Dolphin Place, Holywell Hill, St. Albans, Herts AL1 1EX
Telephone Orders welcome. Tel: 0727 40303