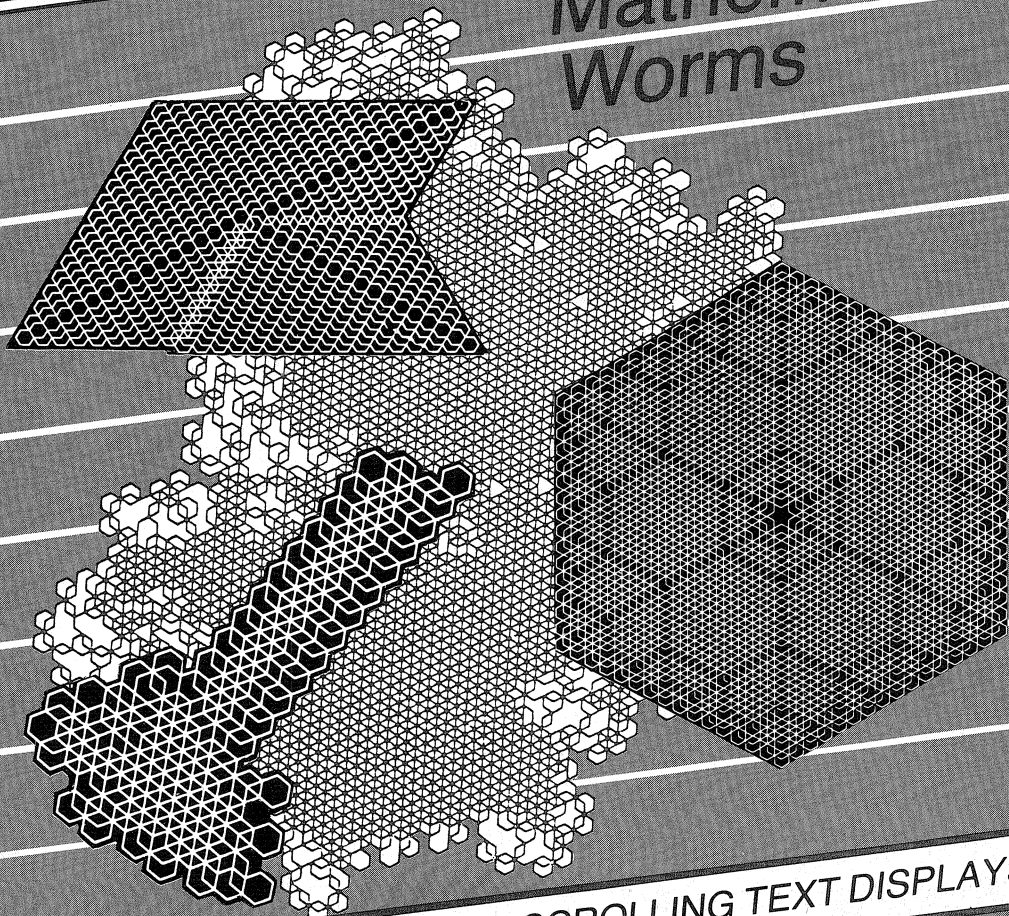


Vol.7 No.4 August/September 1988

BEEBUG

FOR THE
BBC MICRO &
MASTER SERIES

Mathematical Worms



- DUAL SCREEN UTILITY
- SCROLLING TEXT DISPLAYS
- TEMPERATURE MEASUREMENT
- CROSSWORDS

FEATURES

Crossword Editor	8
Running a Temperature	11
Mathematical Worms	14
BEEBUG MiniWimp (Part 3)	19
Viewing Foreign Parts	22
Adventure Games	25
Matrices in Basic (Part 2)	26
First Course -	30
Just Scrolling	42
File Handling for All (Part 4)	48
Using Assembler (Part 2)	52
Workshop -	55
Linked Lists (Part 2)	58
Dual Screen Program Display	62
The Comms Spot	
512 Forum	

REVIEWS

Paintbox and Illustrator	6
Security ROMs Reviewed	37
Font ROM Survey	38
The Bank Manager	46

REGULAR ITEMS

Editor's Jottings	4
News	4
Supplement	33-36
Hints and Tips	61
Postbag	65
Subscriptions & Back Issues	66
Magazine Disc/Tape	67

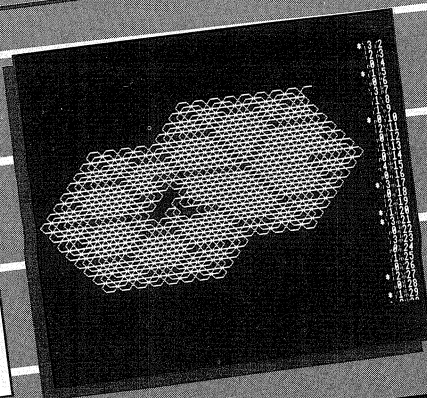
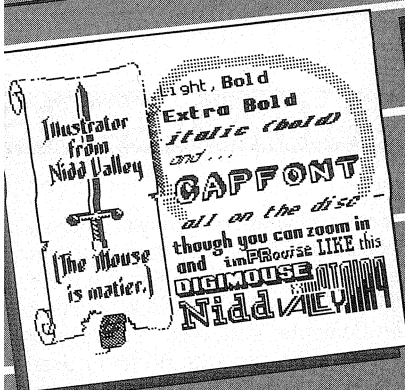
HINTS & TIPS

Master Serial Port	
Telling Basic Apart	
Compact View	
Basic Errors	

PROGRAM INFORMATION

All programs listed in BEEBUG magazine are produced direct from working programs. They are listed in LISTO1 format with a line length of 40. However, you do not need to enter the space after the line number when typing in programs, as this is only included to aid readability. The line length of 40 will help in checking programs listed on a 40 column screen.

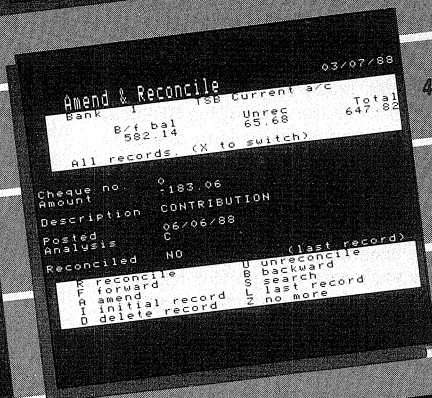
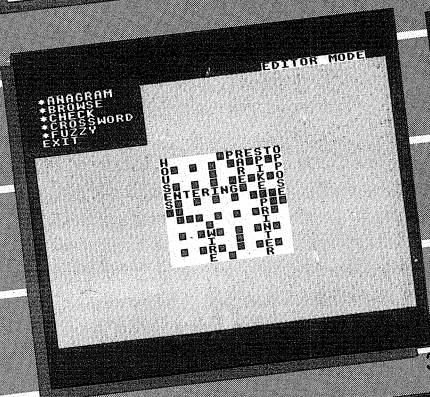
Programs are checked against all standard Acon systems (model B, B+, Master, Compact and Electron; Basic I and Basic II; ADFS, DFS and Cassette filing systems; and the Tube). We hope that the classification symbols for programs, and also reviews, will clarify matters with regard to compatibility. The complete set of icons is given



1. Paintbox and Illustrator

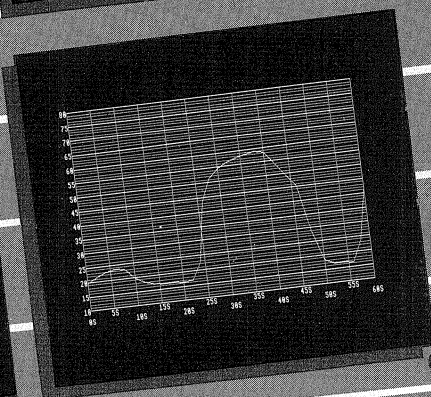
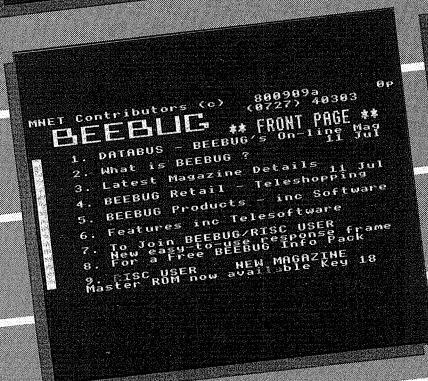
2. Mathematical Worms

3. Crossword Editor



4. Bank Manager

5. Comms Spot



6. Running a Temperature

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 VI)

Model B (Basic II)

Model B (Basic I)

Electron



Filing System

ADFS

DFS

Cassette

Tube Compatibility

Tube



Editor's Jottings

THE MAGAZINE

This month we are publishing the third and concluding part of our series on the BEEBUG Mini-Wimp, our WIMP system for the BBC micro. We are sorry for the fact that we were unable, at very short notice, to publish this as intended in the July issue. This third part discusses how to make the most of the BEEBUG Mini-Wimp and the accompanying icon designer program in generating your own applications based on this system.

We have also changed direction with our series on assembler language programming, now that we have covered most of the basic groundwork. Our follow-up series will concentrate on the application of assembler programming, starting with that all-important area of graphics. If you are interested in exploiting your knowledge of assembler, then you may well find some useful ideas in this section.

Now that the Master is no longer the new machine it was two years or more ago we feel that it is preferable to integrate our coverage of the Master and Compact with the rest of the magazine. We will, of course, continue to feature articles and programs specifically for this range. Both the Master 128 and Master Compact are excellent machines, and Acorn's announcement earlier this year that a further 40,000 Master systems were being ordered from the manufacturers only serves to underline just how good these 8-bit systems still are.

At the same time we have also introduced an occasional series of articles in support of the 512 co-processor and its users. This is in direct response to a good many letters we have received this year asking BEEBUG to help and support users of this system. The frequency and content of future articles on this subject will very much depend on the response we receive from members. So if you have any particular topics or problems you would like dealt with, let us know.

Please note that this issue of BEEBUG covers both August and September. The next issue will be that for October.

News News News N

WAPPING DTP

Watford Electronics is the latest company to produce a Desk Top Publishing (DTP) system for BBC micros. The software, called the 'Wapping Editor' is supplied on a 64K EPROM and uses the PAL-PROM technique pioneered by Computer Concepts. The Wapping Editor is designed to be used with Watford's Quest Mouse, and includes a drawing package. There are features to draw a wide range of shapes, and to manipulate pictures in a variety of ways. A further nice touch is the ability to import and manipulate pictures captured using Watford's video digitiser.

Unlike most other DTP packages, the Wapping Editor includes a versatile word processor. This not only saves having to purchase a second piece of software, but also makes the inclusion of text much easier, because you do not have to switch between programs just to make minor changes. Also included is a comprehensive font designer, and an easy to use page layout utility that allows areas of a page to be defined by dragging a box around the screen. The Wapping Editor should be available in a few weeks time, although the release date and price have not yet been finalised.

PEARTREE SAVED

The Huntingdon based firm Peartree Computers, who we reported in BEEBUG Vol.17 No.2 as being in receivership, has been bought by DRAM Electronics. Stockport based DRAM has taken over all the assets, premises, and name of Peartree. It has been revealed that when Peartree put its business in the hands of the receiver nearly two months ago, DRAM Electronics was one of the major creditors. It is probably this that prompted the take-over deal, which is believed to involve a six-figure sum.

The new company will be known as Peartree-DRAM, and it is expected that it will continue trading along the same lines as Peartree did. All of the existing staff, including former Peartree owner Vartan Mundigian, will be kept on.

News News News News News News News News N

There is still no word as to the future of the Music 87 synthesiser system, which we believe was one of the main reasons that Peartree 'went under'. John Wandells of DRAM confirmed that Peartree did have a large stock of the Acorn Music 500, on which the Music 87 is based. He also said that the company was anxious to get together with Hybrid Technology who produce the rival Music 5000. It is also thought that Peartree-DRAM will be promoting PC clones for educational use.

ARCHIMEDES HARD DISC

Watford Electronics is set to release an alternative to the Acorn upgrade for Archimedes 305 and 310 owners who want to use hard discs. The Watford system will be similar to the official Acorn unit, consisting of 3.5 inch Winchester drive, and a hard disc controller podule, both mounted internally to the computer. A podule backplane must also be fitted to the machine, and this must be purchased separately. It is not yet known how much the Watford add-on will cost, although it is expected to be at least £100 cheaper than the Acorn equivalent, which sells for £575 inc. VAT. Watford Electronics are at Jessa House, 250 Lower High St., Watford WD1 2AN, phone (0923) 37774.

PRESTEL PRICE RISE

British Telecom has increased its subscription and log-on charges for Prestel, which will in turn affect subscribers to Micronet. The actual cost of subscribing to Prestel and Micronet together has risen from £66 to £79.95 a year. Business subscribers will now have to pay £119.95 a year. But the major shock, is a change in the charges for time during which you are actually logged-on. Previously, there was a charge of 6 pence per minute between 8.30 am and 6 pm weekdays and Saturdays mornings, with all other times being free (except for the cost of the phone call). However, an additional charge of 1p a minute has been added for all times except between midnight and 8 am. This may not seem much, but for someone who logs on for an hour twice a week, the call charge is almost as much as the annual subscription.

Things are not all bad though. Telemap, owner of Micronet, has launched Interlink which allows users to access Telecom Gold through Micronet. Telemap have also moved away from Prestel, and started Funtel which is a viewdata system containing a number of entertainment services. This includes the Hotel California system (see news in BEEBUG Vol.6 No.10). Telemap are at Durrant House, 8 Herbal Hill, London EC1R 5ES, phone 01-278 3143.

SOLINET SOFTWARE

Solinet, who are the official user group for owners of Solidisk add-ons, has started a public domain software scheme. The idea of public domain software is that programs, usually of a high standard, can be copied freely and passed on to others at will. The Solinet software will consist of a large number of programs for BBC computers, with special attention paid to software that uses Solidisk products. All the programs will be available on 80 track discs, with the cost being just enough to cover the price of the disc and postage. Solinet also welcome submissions for possible inclusion as public domain software, although they stress that all programs submitted must be copyright free, and be submitted with the knowledge of the original author. More details from Solinet at 13 St John St., Bridlington, East Yorkshire YO16 5NL.

Z88 MODEMS

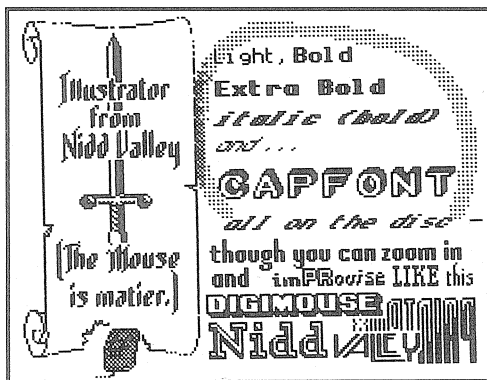
Cambridge Computer has released a pocket modem for use with its Z88 portable computer. The modem is in fact a re-badged version of the Datatronics 1200P which is imported from Taiwan. Both 300/300 and 1200/1200 baud speeds are supported, and while 1200/75 baud is not available, this should not cause any problems as most Prestel nodes now support multi-speed Vascomm systems. The modem, which does not yet have BABT approval, is the size of a cigarette packet, and is powered by a small PP3 battery. The cost is £172.44 inclusive, which also includes a cut down version of Wordmongers' ZTerm terminal software in a cartridge. More details from Cambridge Computer on (0223) 312216. B

PAINTBOX and ILLUSTRATOR

Nidd Valley claims that its latest packages provide desk-top publishing facilities for the BBC micro. How much truth is there in this claim? Roger Burg investigates.

Products	Paintbox and Illustrator
Supplier	Nidd Valley Thorp Arch Trading Estate, Wetherby, West Yorkshire LS23 7BJ. Tel. (0937) 844661
Price	£19.90 inc VAT (Paintbox) £49.00 inc VAT (Illustrator)

Paintbox and Illustrator are mouse-driven packages and use the Nidd Valley mouse, but claim that they work with any other BBC compatible mouse as well. The Illustrator also runs well from the keyboard.



PAINTBOX

Paintbox is an easy-to-use mode 1 painting program. Its response is a little slow, but in practice this is not a problem. It can use any four of the BBC's eight colours. These can be extended further with four pixel mixes. It can draw lines, rectangles, circles, and stretchable but not rotatable ellipses. The rectangles and lines are rubber-banded. It has a limited plain fill, and allows freehand lines in any of six

brush shapes. There are simple facilities for text, cut-and-paste, grid and very rudimentary filing facilities.

Zoom, airbrush and all scrolling are significantly missing as are polygons, patterns, rays and other less valuable features, and the menus permanently obstruct 15% of the screen.

Nice features are a quick mode conversion from mode 4 to mode 1, a menu option for future expansions, an attempt to protect the Break key (ignored on some expensive software), and the idea of saving the screen every five minutes (which is sadly spoiled in its implementation). Most useful is the new Integrex colour printer dump.

On balance, I can find no justification for yet another mode 1 graphics package in this price range. But if you have a mouse and want a fun application this would fill the bill.

ILLUSTRATOR

Illustrator is faster and more comprehensive. It is essentially a simple mode 4 graphics package. The features include the ability to dump the output to any shape or size area of the printer's page. This is particularly well implemented, but it does not begin to justify the extravagant subtitle "Desk Top Publishing Graphics Software". In fairness to the package, I have ignored these claims.

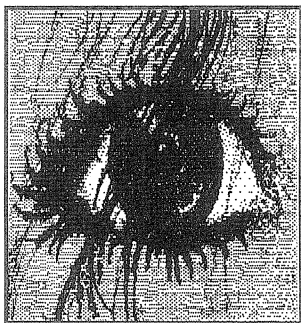
The program comes on two discs, with a 25 page, dot-matrix printed manual. The first disc is only used for starting up and appears to be for protection. This brings up the main screen, and 60% of it is available for drawing.

The main features include all the usual drawing options applicable to a monochrome screen. They are generally well implemented and quick. A few unnecessary extras have been added when more attention should have been paid to essentials.

For instance, a range of regular polygons is provided. Polygons were never very useful anyway. But these are extended to include rubber-banded 'segments' and 'sectors' like the circles and ellipses, as well as the facilities to

stretch and to rotate. Since the precision with which the sizes can be specified is fixed to steps (or strides) of eight pixels, the whole range of shapes has little more than curiosity value, and other problems undermine this group of routines.

Light, bold and extra-bold text is provided. The fonts seem to be derived from the BBC character set. They do not support proportional spacing, but the ability to place and type lines of text on the screen, or delete a whole line and instantly restore its background, is very easy to use. There is also a facility to use 16 by 16 bit-mapped fonts. Again the coding is very competent. Unfortunately, the facility does not really come up to scratch. You can design fonts yourself, but the font designer provided is weak in several respects, and larger sizes cannot do without proportional spacing. The zoom facility allows odd lines of text to be drawn, but this is not what graphics utilities are bought for!



Two pattern-fill routines are provided. The more comprehensive one fails to report whether it has filled completely or just run out of memory, and it misses certain rows. An incomplete fill

is not really acceptable nowadays. Single button-presses can repeatedly undo and instantly re-establish the fill to check it. Both of these are well implemented, but in practice are seldom needed.

The cut and paste options are entirely rectangular, and within this limitation they work well. The source image can come from another screen, it can be copied exactly, or stretched to fit any rectangle the mouse can define, or be crushed and stretched in various predefined proportions. Scrolling features (other than in zoom mode) are significantly missing. The image is 32 by 24 characters and not expandable.

Two sets of patterns or shades are provided. As in most graphics packages, these have been carefully chosen to tantalise rather than be useful (who devises these incompatible shade patterns, and grey scales with vast gaps?). But then there are thirty two pattern options on-screen at once, and you can always create and save your own. The pattern library can be defined for other uses, for example for large text characters or circuit design, though these are not offered entirely seriously.

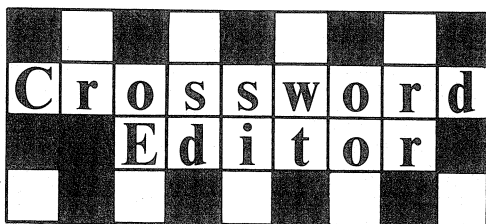
Eight 'brushes' or nib sizes are available on-screen, which can be redesigned and saved, and more are available on disc. Some features work together, so you can 'paint' in any of the patterns with any brush, but line drawing and other features ignore them.

A tiny zoom window helps to line up an area for pixel editing. This works very well, and the screen, magnified eight times, scrolls easily around the full image. As the image is only 256 by 192 pixels, detailed manipulation is essential. Within the zoom you can only set or clear pixels, but in this price range, I think that this is acceptable.

The filing options omit star commands. This is a serious omission. You cannot even rename, delete or catalogue a disc. The program tries to shield the user from these complexities. You can only load files which are presented on the appropriate loading menu, and you can't accidentally overwrite an existing file (but then it also checks if you wish to overwrite a file that you've asked to load, which rather destroys the effect). I would prefer the option to save compacted data rather than the well intentioned hand-holding.

Extended options include several printer dumps, two of which can be your own. None of those provided is really high resolution, though full scaling facilities allow you to select the rectangular shape and size you want on the page. This is done particularly nicely, presenting the user with an image of the printer page (complete with sprocket holes), and the outlined shape of the area to be filled, with the co-ordinates also represented in printer

Continued on page 21



Keith Sumner's crossword editor will appeal to both crossword solvers and designers alike. And as a bonus, it can optionally access SpellMaster for anagrams and other crossword aids.

The Crossword utility presented here is both a crossword designer and editor. Use it first to design the crossword (or copy an existing one), and then fill in the answers. Enter the listing as printed and save it away. When the program is run it will prompt the user for the size of the crossword to be worked on (this must be an odd number). The program then enters design mode in which a blank design area is displayed. Move the flashing cross cursor around the design area with the cursor keys, and use f0 and f1 to delete and place 'blocks' on the grid to specify the crossword design. Other function key options are available as follows.

DESIGN MODE FUNCTION KEYS

- f0 - Delete block
- f1 - Set block
- f2 - Access to Spellmaster ROM
- f3 - Clear design and restart
- f4 - Cross cursor ON
- f5 - Cross cursor OFF
- f6 - Save crossword
- f7 - Load crossword
- f8 - Toggle symmetry checking ON/OFF
- f9 - Exit design mode

When option f9 is selected the program performs a check to see that the crossword design has the correct symmetry. If this criterion is not satisfied by the design, the user is returned to design mode, and the crossword must be edited further to give it the correct symmetry. Only when the design is correct will editor mode be entered. Note that it is possible to toggle on and off the check for symmetry with f8.

In editor mode the grid is again accessed using the cursor keys. Typing a letter will enter/overwrite the letter at the cursor position on the grid, provided the cursor is not on a 'blocked out' square. The Delete key is used to erase an incorrect letter. Additional function key operations are available in editor mode as well.

EDITOR MODE FUNCTION KEYS

- f2 - Access to Spellmaster ROM
- f4 - Cross cursor ON
- f5 - Cross cursor OFF
- f6 - Save crossword
- f7 - Load crossword
- f9 - Exit editor mode and end program

The SpellMaster ROM from Computer Concepts provides a useful means of helping with the solution of crosswords. For those with this ROM fitted, the Crossword program allows direct access to many of SpellMaster's facilities. If you do not have SpellMaster, just ignore these features.

When f2 is selected, the user is presented with a window of options. To select an option, move up and down the list with the cursor keys and use Return to select the option. Option 6 is the exit option and returns the user to the design/editor mode. Options 1 to 5 then prompt the user for a search string. Pressing Return will exit string-entry mode and no search occurs. Entering a string will pass control to SpellMaster before returning you to design/editor mode.

The program is well structured and easy to follow. Notice that a small machine code routine in PROCmc is used to de-tokenise Basic lines as they are needed.

The powerful facilities offered by this editor in conjunction with SpellMaster make the design of quite complex crosswords much simpler, and will make a good addition to your software library.

```

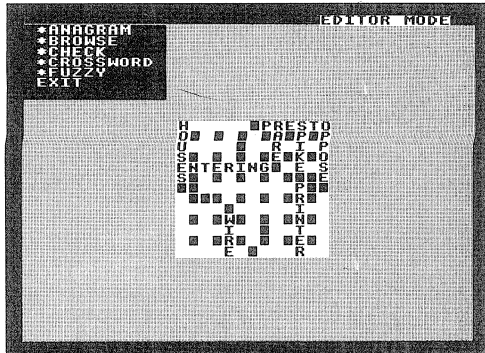
10 REM Program      Crossword
20 REM Version      B1.02
30 REM Author       Keith Sumner
40 REM BEEBUG       August 1988
50 REM Program      subject to copyright
60 :
100 ON ERROR GOTO 120
    
```



```

110 *ANAGRAM
120 IF ERR=3 THEN spell=TRUE ELSE spel
l=FALSE
130 :
140 ON ERROR PROCend:REPORT:PRINT" at
line";ERL:END
150 MODE1:PROCsetup
160 PROCdesign:PROCsolve:PROCend:END
170 :
1000 DEF PROCend
1010 *FX229,0
1020 *FX4,0
1030 VDU26,30:ENDPROC
1040 :
1050 DEF PROCwarn
1060 PRINTTAB(1,1)"Crossword not symmet
rical":PROCcont:ENDPROC
1070 :
1080 DEF PROCon:VDU23,1,1,0;0;0;:ENDPRO
C
1090 DEF PROCoff:VDU23,1,0;0;0;:ENDPR
OC
1100 :
1110 DATA *ANAGRAM,*BROWSE,*CHECK,*CROS
SWORD,*FUZZY,EXIT
1120 :
1130 DEF PROCsetup
1140 *FX229,1
1150 *FX4,1
1160 *FX225,140
1170 VDU23,224,24,24,24,195,195,24,24,2
4:VDU23,255,126,126,126,126,126,126,
0
1180 CLS:INPUT" Enter size of crossword
:D%"
1190 IF D%<=1 ORD%>27 OR D% MOD2=0 PRIN
T" Invalid size":PROCcont:GOTO1180
1200 VDU12,19,0,4;0;19,3,0;0;19,2,7;0;
1210 DIMW% D%^2,OS(5)
1220 X=1:Y=1:70=D%:RESTORE1110
1230 U%=19-D%DIV2:V%=15-D%DIV2
1240 FOR X% =0 TO 5:READO$(X%):NEXT
1250 dochk=TRUE:PROCclear:PROCmc
1260 ENDPROC
1270 :
1280 DEF FNget (A%,B%) =W%?FNpos (A%,B%)
1290 DEF PROCput (A%,B%,C%)
1300 W%?FNpos (A%,B%) =C%:ENDPROC
1310 DEF FNpos (A%,B%) =D%*(A%-1)+B%-1
1320 :
1330 DEF PROCclear
1340 FORX%=0TOD%*2:W%?X%=0:NEXT:ENDPROC
1350 :
1360 DEFFNmode (N%)
1370 IFN%=0 ="DESIGN" ELSE ="EDITOR"
1380 :
1390 DEF PROCchk

```



```

1400 C$="Disabled":COLOUR129:COLOUR2
1410 IF dochk C$="Enabled "
1420 PRINTTAB(25,1)"Symmetry Check"
1430 PRINTTAB(27,2)C$:COLOUR130
1440 ENDPROC
1450 :
1460 DEF PROCdesign
1470 REPEAT:on=TRUE:exit=FALSE
1480 mode$=FNmode(0):PROCdraw
1490 REPEAT
1500 A=INKEY(2):PROCchar(32)
1510 IF A>135 ANDA<140 PROCmove
1520 IF A=140 PROCput(X,Y,0)
1530 IF A=141 PROCput(X,Y,1)
1540 IF A=142 AND spell PROCspell:PROCd
raw
1550 IF A=143 PROCclear:PROCdraw
1560 IF A=144 on=FALSE
1570 IF A=145 on=TRUE
1580 IF A=146 PROCsave
1590 IF A=147 PROCload
1600 IF A=148 dochk=NOTdochk:PROCchk
1610 IF A=149 exit=TRUE
1620 IF on chr=224 ELSEchr=0
1630 PROCchar(chr)
1640 UNTILexit
1650 IF dochk ok=FNcheck ELSEok=TRUE
1660 IF NOT ok PROCwarn
1670 UNTIL ok
1680 ENDPROC
1690 :
1700 DEF FNcheck:LOCALX%,Y%,d%,e%
1710 nosym=FALSE:d%=1+D%DIV2
1720 e%=2*d%:X%=0
1730 REPEAT:X%=X%+1:Y%=0
1740 REPEAT:Y%=Y%+1
1750 IF FNget(X%,Y%)>1 GOTO1790
1760 S1=FNget(X%,Y%):S2=FNget(e%-X%,e%-
Y%)
1770 S3=FNget(e%-Y%,X%):S4=FNget(Y%,e%-
X%)

```

```

1780 nosym=NOT(S1=S2 AND S2=S3 AND S3=S
4)
1790 UNTIL Y%=D% ORnosym
1800 UNTIL X%=D% ORnosym
1810 =NOTnosym
1820 :
1830 DEF PROCsolve
1840 mode$=FNmode(1):PROCdraw:PROCwords
1850 ENDPROC
1860 :
1870 DEF PROCchar(Z)
1880 LOCALA%,x%,y%,c%,fgd%
1890 x%=X+U+:y%=Y+V+:fgd%=0:c%=Z:A%=FNg
et(X,Y)
1900 IF A%=1 ANDZ=32 fgd%=3:c%=255
1910 IF A%>64 ANDZ=32 fgd%=0:c%=A%
1920 COLOUR fgd%:VDU31,x%,y%,c%
1930 ENDPROC
1940 :
1950 DEF PROCmove
1960 IF A=137 ANDX<D% X=X+1
1970 IF A=136 ANDX>1 X=X-1
1980 IF A=139 ANDY>1 Y=Y-1
1990 IF A=138 ANDY<D% Y=Y+1
2000 ENDPROC
2010 :
2020 DEF PROCfill
2030 COLOUR0:COLOUR130
2040 PRINTTAB(26,0)mode$;" MODE":CALL&A
00
2050 ENDPROC
2060 :
2070 DEF PROCdraw
2080 COLOUR128:VDU12:PROCOff
2090 VDU24,32;32;1248;992;;GCOL0,129,16
2100 IF mode$="DESIGN" PROCchk
2110 PROCfill:ENDPROC
2120 :
2130 DEF PROCwords
2140 X=1:Y=1:on=TRUE:exit=FALSE
2150 REPEAT:A=INKEY(2):PROCchar(32)
2160 IF A>135 ANDA<140 PROCreplace
2170 IF A>96 ANDA<123 A=A-32
2180 IF A>64 ANDA<91 PROCletter
2190 IF A=127 PROCerase
2200 IF A=142 AND spell PROCspell:PROCd
raw
2210 IF A=144 on=FALSE
2220 IF A=145 on=TRUE
2230 IF A=146 PROCsave
2240 IF A=147 PROCload:PROCverify
2250 IF A=149 exit=TRUE
2260 IF on chr=224 ELSEchr=0
2270 PROCchar(chr)
2280 UNTIL exit:PROCreplace
2290 ENDPROC
2300 :

```

```

2310 DEF PROCverify
2320 ok=FNcheck:IF ok ENDPROC
2330 PROCwarn:PROCdesign
2340 mode$=FNmode(1):A=0:PROCdraw
2350 ENDPROC
2360 :
2370 DEF PROCletter
2380 IF FNget(X,Y)=1 ENDPROC
2390 PROCchar(A):PROCput(X,Y,A)
2400 ENDPROC
2410 :
2420 DEF PROCerase
2430 IF FNget(X,Y)>1 PROCput(X,Y,0)
2440 PROCchar(32):ENDPROC
2450 :
2460 DEF PROCreplace
2470 C%=FNget(X,Y):IFC%>64 PROCchar(C%)
2480 PROCmove:ENDPROC
2490 :
2500 DEF PROCsave
2510 PRINTTAB(1,1)"SAVE CROSSWORD"
2520 PROCon
2530 INPUTTAB(1,2)"Filename : "save$
2540 IF save$="" GOTO2530
2550 save=OPENOUT(save$):PRINT#save,D%
2560 FORX%=1TOD%:FORY%=1TOD%
2570 BPUT#save,FNget(X%,Y%)
2580 NEXT:NEXT
2590 CLOSE#save:PROCOff:PROCdraw
2600 ENDPROC
2610 :
2620 DEF PROCload
2630 PRINT TAB(1,1)"LOAD CROSSWORD"
2640 E%=D%:PROCon
2650 INPUTTAB(1,2)"Filename : "load$
2660 IF load$="" GOTO2750
2670 load=OPENUP(load$)
2680 IF load=0 PROCnofile:GOTO2750
2690 PROCclear:INPUT#load,D%
2700 FORX%=1TOD%:FORY%=1TOD%
2710 PROCput(X%,Y%,BGET#load)
2720 NEXT:NEXT:CLOSE#load
2730 U%=19-D%DIV2:V%=15-D%DIV2:??&70=D%
2740 IF D%>E% PROCstop:END
2750 PROCOff:PROCdraw
2760 ENDPROC
2770 :
2780 DEF PROCstop
2790 PROCend
2800 VDU7:CLS:PRINT"Dimension of new pu
zzle is""greater than previous puzzle"
2810 ENDPROC
2820 :
2830 DEF PROCnofile
2840 PRINTTAB(1,3)"File not found"
2850 PROCcont:ENDPROC
2860 :

```

Continued on page 57

RUNNING A TEMPERATURE



Bernard Hill describes a temperature probe which may be connected to the Beeb's analogue port, and calibrated to give accurate results both cheaply and easily.

The idea of a temperature probe for the BBC is not new, but the beauty of this implementation is its simplicity and low cost. Maplin Electronics sell a range of assembled temperature sensors and kits. Separately available for use with these kits are two sensor probes, which cover a range of -40°C to 110°C . They each consist of a temperature-sensitive variable resistor moulded into a plastic, metal-capped stylus, attached to a length of about 3 metres of twin-core wire. All that is needed in addition to the probe is a resistor and a 15-pin D-connector (see Diagram 1). Diagram 2 shows the pin connections to the D-socket viewed from outside the Micro's case. The value of the resistor is not crucial but should be around 2000-5000 ohms.

The complete shopping list from Maplin is given below.

	Price	Cat No
Temperature probe		
-40° to 50°C	£1.95	FP65V
20° to 110°C	£1.95	FP66W
D-plug	£0.58	BK58N
D-plug cover	£0.90	BK60Q
3300 ohm 1/8 watt resistor	£0.03	U3K3
Postage	£0.50	
Surcharge (order below £4.50)	£0.50	
TOTAL	£4.46	

You will also need a low power soldering iron and solder in order to connect the resistor and plug. Take care here not to use excess solder which might short a connection, nor to use too much heat and melt the insulation around the wire. Assembly is very straightforward if you follow Diagram 2. Note that you can leave the resistor within the body of the plug before screwing on the cover.

Because the precise value of the resistor is not known, you will need to calibrate the probe before use. This is the purpose of the Probe Calibration program, but before you run this you will need to have a normal thermometer ready and two to five constant temperature sources. I use a mercury thermometer and three or four glasses of water at

temperatures between cold and about 50°C. Run the program and place the probe and the reference thermometer in the coldest beaker. After a few seconds press the Space Bar. The program is now sampling and averaging the value of analogue channel 1. When the Space Bar is released you must enter the value on the reference thermometer to complete one calibration point.

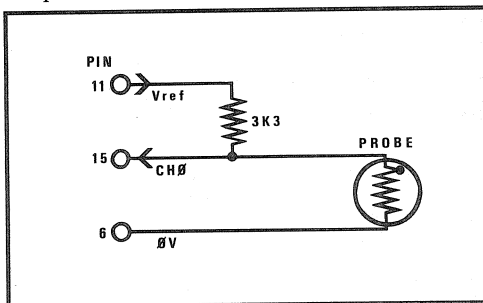


Diagram 1

Now transfer the probe and thermometer into a second beaker and repeat, but in order to allow the thermometer to stabilise its reading I suggest that you wait for a minute or so before pressing the Space Bar to sample the probe again.

Do this for at least two calibration points and press Escape to perform the final calibration calculation. If you have used at least three sources then a correlation coefficient is quoted by the program to assess the degree of linearity of the calibration points. This should be very close to 1, at least 0.99. A function DEFFNtemp is then listed on the screen and appended automatically to the program as line 32767. It can be tested with PRINT FNtemp and of course will be in the same units as your calibration thermometer (Centigrade or Fahrenheit). Make a note of the coding for DEFFNtemp as you will need it for any program which uses the temperature probe.

The bother of calibrating the probe is offset by the fact that it only needs to be done once. Your version of FNtemp will always be the same from now on. To increase your accuracy you could average your subsequent temperature readings over at least one second. This is an example procedure that uses FNtemp to average a reading over a second.

```

10000 DEFFNtemp2
10001 LOCAL S,N,T:T=TIME:S=0:N=0
10002 REPEAT S=S+FNtemp:N=N+1
10003 UNTIL TIME-T>100
10004 =S/N

```

The Probe Plot program is a simple but versatile graph drawing program which plots the output of

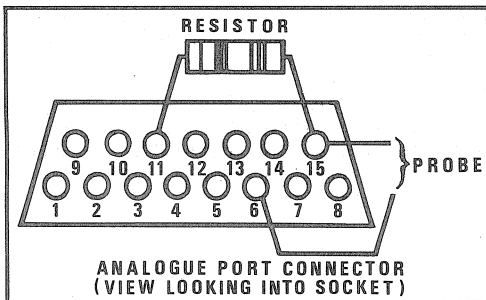


Diagram 2

the probe over any period from a few seconds to several weeks. With this program you could plot the temperature in any school science experiment, or graph the outside temperature over a complete month or more. Printer dumps or *SPOOL statements for a Mode-00 type dump (see BEEBUG Vol.6 No.2 Page 50 and No.6 page 52) can be added at the lines shown in the listing. When run, the program requests a temperature range and a time period for plotting. It also requests a sampling interval over which it will average the temperature probe reading. The longer you make this the more accurate your graph, and the fewer points will go to your *SPOOL file (if you are using one). When you are ready to start sampling, press any key and watch the graph build up. Press any key at the end to finish.

The possibilities for this probe are endless, both at home and in the classroom. Using the function FNtemp, the probe may be used easily within your own software.

Maplin Electronic Supplies Ltd can be reached at PO Box 3, Rayleigh, Essex, or 0702-554161 for credit card orders with next day delivery.

```

10 REM Program  Probe Calibration
20 REM Version  B1.12
30 REM Author   Bernard Hill
40 REM BEEBUG   August 1988
50 REM Program  Subject to copyright
60 :
100 ON ERROR GOTO 430
110 Lim=10
120 MODE7:PROctitle2("Temperature Cali
bration",132,135)
130 VDU28,0,24,39,5
140 DIM adval(Lim),temp(Lim):M%=0
150 REPEAT
160 PRINT"Press space bar to record t
emperature"TAB(7)"or Escape to calculat
e:"

```

```

170 REPEAT UNTIL INKEY=99
180 N%=0:S%=0:REPEAT
190 PRINT"Calibrating ...":VDU11
200 S%=S%+ADVAL1:N%=N%+1
210 UNTIL INKEY=99=0
220 *fx15 1
230 M%=M%+1:adval(M%)=S%/N%
240 PRINT""Temperature during this ti
me:";
250 INPUT ""temp(M%):UNTIL M%=Lim
260 @%=&90A
270 IF M%<2 PRINT"Not enough points":V
DU26,31,0,24:END
280 MODE7:PRINT"Calculation based on "
;M%:" points:";
290 a=0:a2=0:t=0:t2=0:p=0
300 FOR i=1 TO M%
310 a=a+adval(i):a2=a2+adval(i)^2
320 t=t+temp(i):p=p+adval(i)*temp(i)
330 t2=t2+temp(i)^2
340 NEXT
350 avt=t/M%:ava=a/M%:vp=p/M%-a*t/M%^2
360 va=a2/M%-(a/M%)^2:vt=t2/M%-(t/M%)^
2
370 g=vp/va:t0=avt-g*ava
380 IF M%>2 THEN PRINT"Correlation: ";
-vp/SQRva/SQRvt;
390 s$="32767DEFNtemp="+STR$t0+"-ADVA
L1/"+STR$(1/g)
400 OSCLI("KEY0 "+s$+"|M"): *fx138,0,12
8
410 END
420 :
430 IF ERR=17 GOTO 260
440 REPORT
450 PRINT" at line "ERL
460 END
470 :
1000 DEF PROctitle2(t$,c1,c2)
1010 LOCAL i,a$:a$=CHR$c1+CHR$c157+CHR$c
2
1020 PRINTa$:FOR i =1 TO 2
1030 PRINTa$:CHR$141;TAB(20-LENT$/2);t$
1040 NEXT:PRINTa$:ENDPROC

```

```

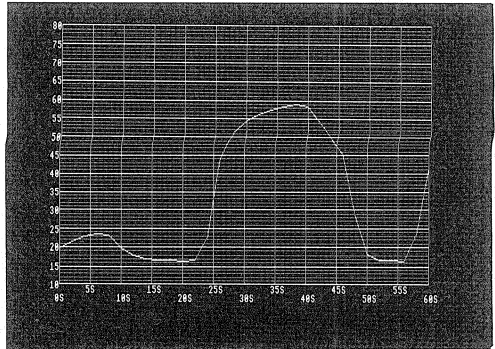
10 REM Program  Probe Plot
20 REM Version  B1.13
30 REM Author   Bernard Hill
40 REM BEEBUG   August 1988
50 REM Program  Subject to copyright
60 :
100 ON ERROR GOTO 250
110 top=1199:H=940:MODE7:DIM mul(5)
120 mul(1)=1:mul(2)=60:mul(3)=3600
130 mul(4)=86400:mul(5)=604800
140 PROctitle2("Temperature Graph",132
,131)
150 PROOptions
160 REM insert *SPOOL filename here fo

```

```

r MODE=00 dump
170 MODE0:PROCaxes
180 PROCdraw
190 REM insert screen dump or *SPOOL h
ere
200 *FX15,1
210 IF GET
220 MODE7
230 END
240 :
250 MODE7
260 REPORT:PRINT" at line "ERL
270 END
280 :
1000 DEF PROCTitle2(t$,c1,c2)
1010 LOCAL i,a$:a$=CHR$c1+CHR$157+CHR$c
2
1020 PRINTa$:FOR i =1 TO 2
1030 PRINTa$,CHR$141;TAB (20-LENT$/2);t$
1040 NEXT:PRINTa$:ENDPROC
1050 :
1060 DEF PROCOptions
1070 LOCAL @%:@%=&1020104
1080 REPEAT
1090 PRINTTAB (10,5) "Temperature Range:"
,
1100 PRINTTAB (0,7) "Minimum :";SPC11;"Ma
ximum :";SPC11;TAB (9,7);
1110 INPUT"m$:minT=VALm$
1120 PRINTTAB (29,7);
1130 INPUT"m$:maxT=VALm$
1140 UNTIL maxT>minT
1150 REPEAT
1160 tspan=FNtime("Time Span:",9)
1170 U1=U2:U1$=U2$
1180 tincr=FNtime("Sample Interval",15)
1190 UNTIL tspan>=2*tincr AND tincr>=1
1200 PRINTTAB (0,19) :PROCTitle2("Current
temperature:",132,135)
1210 PRINTTAB (10) "Any key to start";
1220 REPEAT S1=0:N%=0:T=0:M%=0:T%=TIME
1230 REPEAT S1=S1+FNtemp:N%=N%+1
1240 UNTIL TIME-T%>100:S1=S1/N%
1250 T=T+S1:M%=M%+1:T$=STR$S1
1260 PRINTTAB (32,21) T$;TAB (32,22) T$;
1270 UNTIL INKEY1<>-1:t0=T/M%:ENDPROC
1280 :
1290 DEF FNtime(t$,L):LOCAL m$,t
1300 REPEAT:PRINTTAB (10,L)t$'
1310 PRINTTAB (0,L+2) "Size :
";SPC10
1320 PRINTTAB (0,L+3) "Units (S/M/H/D/W)":
";SPC10
1330 PRINTTAB (20,L+2);:INPUT "m$
1340 t=VALm$:PRINTTAB (20,L+3);
1350 INPUT "U2$:IF U2$="" THEN U2$="s"
1360 U2=(INSTR("SsMmHhDdWw",U2$)+1) DIV
2
1370 t=t*mul (U2):UNTIL t>=1:=t
1380 :
1390 DEF FNy (T)=(T-minT)/(maxT-minT)*H

```



```

1400 DEF FNx(t)=t/tspan*(top+1)
1410 :
1420 DEF PROCaxes
1430 LOCAL @%,t:@%=&10000401
1440 VDU5,29,32;72;
1450 FOR t=minT TO maxT
1460 FOR t=minT TO maxT:y=FNy(t)
1470 IF y<0 OR y>H THEN 1510
1480 IF t MOD 5=0 THEN MOVE -32,y+10:PR
INTt;
1490 MOVE 0,y
1500 IF t MOD 5=0 THEN PLOT 5,top,y ELS
E PLOT 21,top,y
1510 NEXT
1520 line=FNline(tspan):e=FALSE
1530 FOR t=0 TO tspan STEP line
1540 x=FNx(t):MOVEx,0:DRAWx,H
1550 S2=t/mul (U1)
1560 IF e THEN MOVE x-16,-8 ELSE MOVE x
-16,-40
1570 PRINTS2;U1$;e=NOT e:NEXT:ENDPROC
1580 :
1590 DEF PROCdraw
1600 TIME=0:MOVE 0,FNy(t0)
1610 REPEAT S1=0:N%=0:T%=TIME
1620 REPEAT S1=S1+FNtemp:N%=N%+1
1630 UNTIL TIME DIV (100*tincr)>T% DIV
(100*tincr)
1640 T=S1/N%:DRAW FNx(TIME/100),FNy(T)
1650 UNTIL TIME/100>tspan:ENDPROC
1660 :
1670 DEF FNline(t)
1680 IF t>=4320000 THEN =604800
1690 IF t>=432000 THEN =86400
1700 IF t>=86400 THEN =21600
1710 IF t>=21600 THEN =3600
1720 IF t>=3600 THEN =900
1730 IF t>=900 THEN =300
1740 IF t>=300 THEN =30
1750 IF t>=120 THEN =10
1760 IF t>=30 THEN =5
1770 IF t>=10 THEN =1
1780 =1
1810 :

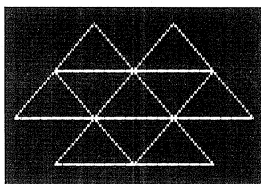
```

B

S MATHEMATICAL WORMS

by Grimble Gromble

From time to time an apparently simple piece of mathematical analysis, implemented on a computer and represented visually captures the imagination of large numbers of people to the point of developing a cult following. Examples that come readily to mind are John Conway's game of Life, and more recently the Mandelbrot set. Now another fascinating world can be revealed in the study and analysis of so-called *Mathematical Worms*.



Mathematical Worms are not new, and were comprehensively described in Scientific American for October 1973. The rules described in that article form the

basis of the program listed here. We will first of all explore the basic rules governing our mathematical worms, and then look in detail at the use of the program which will allow us to explore their intriguing world.

The worm starts from an arbitrary point on an infinite grid of lines. We shall only consider isometric worms, that is worms which traverse a grid of equilateral triangles. As the worm moves it 'eats' the line it is travelling along, so that each line of the grid can be followed just once. When a worm reaches a junction point or node, a *rule* will determine which way it turns. Initially, when a worm reaches a node, there are five potential exit routes (it cannot leave by the route it arrives). If the rules cause the worm to revisit any node, the number of choices becomes more limited. Ultimately, if the worm reaches a node from which there is no exit (either literally or because the rule applying to that node prohibits any exit) the worm dies.

A visual representation will show the route followed by a worm under a given set of rules. Some pathways are quite short, while some are infinite. Some exhibit a regular pattern while others are more random in their appearance. It

is also possible to classify worms into groups or *species* according to the sets of rules which they follow.

Altogether there are 31 different node states depending on the number of exit paths (choices) still remaining, and their relationship to the entry path of the worm. Wherever there is a choice *you* can determine which rule should be used. Choices can be grouped into four *fields* depending on the number of uneaten paths remaining (1, 2, 3, or 4) on exit. This leads to 1296 different sets of rules, so you can see there is plenty of scope here.

USING THE PROGRAM

The program should be typed in and saved as usual. It is very tight on space on a model B (PAGE at &1900), so do not add any extra spaces, and omit the space between the line number and the first instruction which is included for readability.

Running the program will put an initial worm through its paces. This worm simply takes the right-most path at every node and doesn't last long. You will see the worm's route on the screen, and down the right-hand side are shown the rules followed by the worm. Each line of this display looks similar to:

* 1 3 K 7

First, an asterisk is displayed if this particular rule was encountered. The next character shows

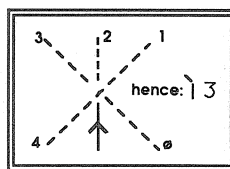


Figure 1

The final character and number (in the range 0 to 30) give the node type and number (that is the uneaten paths and their relationship to the entry path). The number is generated from a binary format indicating the state of the node (read clockwise) as in Figure 2. Nodes 0 to 30 are displayed with their currently selected rules. Node 31 causes the worm to die, so no rule is associated with it. Instead the number of paths traversed is

displayed plus an asterisk if the worm does die by this means, rather than just reaching the edge of the screen display.

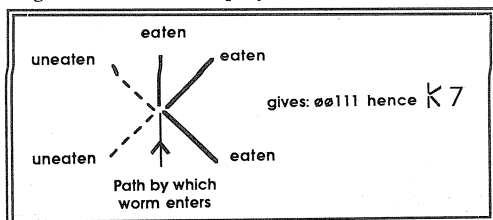


Figure 2

Changing the rule associated with a particular node type will allow you to generate a huge variety of different worm patterns on the screen.

CONTROLLING THE PROGRAM

From the main display the following controls are available.

Up and down cursor keys allow the node type to be selected.

Left and right cursor keys alter the rule for the selected node.

Copy Switches to an *Alternative* display.

W Sets off a worm.

F Also starts a worm, but sets a breakpoint to halt the worm at the first node encountered where a selection of rules is possible.

P Sets pause so that a worm pauses as soon as it has started.

C Clears pause mode.

S Saves the current display, rules and conditions to a file.

L Loads a previous saved display from a file.

H Holds the current set of rules in memory.

R Recalls the previously held set of rules.

The *Alternative* display shows the starting conditions and breakpoints. These have the following meanings.

S: Scale - this is a figure in the range from 1 to 10 which determines the lengths of the line segments in a worm's path. The smallest scale (1) is needed for the longer lasting worms, but even this won't cope with the paths of all worms.

D: Direction in which a worm sets out initially.

X: X co-ordinate of a worm's starting point.

Y: Y co-ordinate of a worm's starting point.

N: Nodal breakpoint. A value other than 0 will cause the worm to pause whenever it encounters a node of this type.

M: Modulus breakpoint. A value other than 0 will cause the worm to pause whenever it has traversed a number of paths equal to a multiple of this number.

Judicious use of D, X and Y can enable some worms to be displayed at a larger scale. The following keys can also be used from the *Alternative* display.

Up and down cursor keys select the condition to alter.

Left and right cursor keys alter the selected condition.

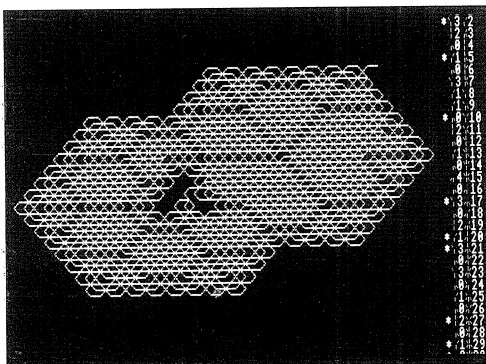
The Shift and/or Ctrl keys may be used in conjunction with these.

Copy Switches back to the main display.

W, F, P, C, L & S operate as from the main display.

Z Sets all of D, X and Y to zero.

E Erases the breakpoints N and M, and switches off pause mode.

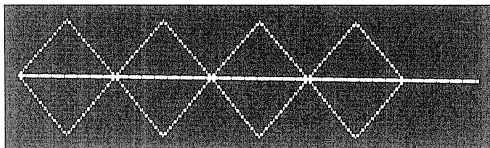


CLASSIFICATIONS

One of the fascinations of the world of mathematical worms, apart from the obvious visual appeal of the paths themselves, is the task of classifying worms according to the set of rules each is governed by. This can be an involved subject so we will only give an introduction here.

One approach to classifying our worms, and an alternative to the one given in the original Scientific American article, is to use the F option and let a worm classify itself. All that is necessary is to write down the list of rules in the order in which they are first encountered.

Consider, as an example, the programmed worm (the one which appears when the program first starts).



Press F and the worm will pause at node 0 for which the rule is set to 0. So 0 starts the classification list. Press C to continue. The worm then stops at node 1 for which the rule is set to 1, extending the list to 01. Press C again. The next pause is at node 3 for which the rule is 2, and so the list becomes 012. Now press C again. The worm next pauses at node 7 for which the rule is set to 3, so the list becomes 0123. Press C and the worm continues without any further pauses until it terminates, so the complete classification of this worm is 0123.

You will see that the worm also visited node type 15 but did not pause at that node as no choice was available. The classification of worms is determined by the choice of rule for those nodes where choice is possible. The same method may also be used to enter the rules to achieve a particular classification. At each pause set the rule as indicated by the appropriate digit in the classification before continuing.

There is much more that could be said about classifying worms, but we will leave you to explore this fascinating world for yourself. We have provide some illustrations of some worms we have generated to spur you on. If you discover anything of particular interest then let us know.

```

10 REM Program Worm
20 REM Version B1.2
30 REM Author Grimble Gromble
40 REM BEEBUG Aug/Sept 1988
50 REM Program subject to copyright
60 :
100 PROCchar: CLEAR
110 PROCinit: A$="W"
120 ON ERROR MODE 7: PROCerror: A$="E"
130 MODE 0: PROCscreen
140 REPEAT
150 IF A$ <> "E" THEN PROCworm(D%, X%, Y%,
A$="F", N%, M%)

```

```

160 A$=FNOption
170 UNTIL FALSE
180 :
1000 DEF PROCinit
1010 c$=CHR$135: l$=CHR$136: r$=CHR$137
1020 d$=CHR$138: u$=CHR$139: cl$=-26
1030 cr$=-122: c$=-83: e$=-35: p$=-56
1040 q$=-17: sp$=-99: shift$=-1: ctrl$=-2
1050 R%=0: R2%=0: N%=0: M%=0: P%=FALSE
1060 mx%=582: my%=508: dx%=6: dy%=4: ms%=10
1070 DIM x%(5), y%(5), c 31, u 31, h 31
1080 READ S%, D%, X%, Y%
1090 FOR I%=0 TO 5
1100 READ x%(I%), y%(I%)
1110 x%(I%)=x%(I%)*S%: y%(I%)=y%(I%)*S%
1120 NEXT
1130 FOR I%=0 TO 31: READ h?I%: NEXT
1140 PROCreset
1150 *FX4, 1
1160 *FX228, 128
1170 ENDPROC
1180 :
2000 DEF PROCscreen
2010 REM TEXT ON LEFT
2020 REM VDU 24, 112; 0; 1279; 1023;
2030 REM VDU 29, 112+mx%, my%;
2040 REM VDU 28, 0, 31, 6, 0
2050 REM TEXT ON RIGHT
2060 VDU 24, 0; 0; 1167; 1023;
2070 VDU 29, mx%, my%;
2080 VDU 28, 73, 31, 79, 0
2090 ENDPROC
2100 :
3000 DEF PROCworm(D%, X%, Y%, F%, N%, M%)
3010 FOR I%=0 TO 31: u?I%=0: NEXT
3020 EX%=mx%-S%*dx%: EY%=my%-S%*dy%
3030 T%=0: CLG: PROCshow: MOVE X%, Y%
3040 REPEAT
3050 X%=X%+2*x%(D%): Y%=Y%+2*y%(D%)
3060 DRAW X%, Y%: T%=T%+1
3070 IF ABS(X%)>EX% OR ABS(Y%)>EY% THEN
C%=31: GOTO 3150
3080 C%=FNnode
3090 IF INKEY(e%) THEN F%=0: N%=0: M%=0: P%
=FALSE
3100 IF F% THEN IF u?C%=0 THEN IF NOT (
C%=15 OR C%=23 OR C%=27 OR C%>=29) THEN
P%=TRUE
3110 IF N% THEN IF N%=C% THEN P%=TRUE
3120 IF M% THEN IF T%MODM%=0 THEN P%=TR
UE
3130 IF INKEY(p%) THEN P%=TRUE
3140 IF P% THEN PROCpause
3150 u?C%=1: D%=(D%+c?C%+4)MOD6
3160 UNTIL C%=31 OR INKEY(q%)
3170 ENDPROC
3180 :
4000 DEF PROCpause

```

```

4010 LOCAL Q%,RS%,RC%,OX%,OY%:@%=6
4020 OX%=X%-2*x%(D%):OY%=Y%-2*y%(D%)
4030 VDU 23,1,0;0;0;0;
4040 FOR I%=0 TO 30
4050 IF u?I% THEN PRINTTAB(0,I%)"*";
4060 NEXT
4070 PRINTTAB(0,31)T%;
4080 PRINTTAB(0,C%);
4090 VDU 23,1,1;0;0;0;
4100 REPEAT
4110 GCOL 0,0:MOVE OX%,OY%:DRAW X%,Y%
4120 IF u?C%=0 AND C%<>31 GOSUB 4190
4130 GCOL 0,1:MOVE OX%,OY%:DRAW X%,Y%
4140 IF INKEY(sp%) Q%=RS% ELSE RS%=TRUE
4150 IF INKEY(e%) THEN F%=0:N%=0:M%=0:P
%=FALSE:Q%=TRUE
4160 IF INKEY(c%) THEN P%=NOT RC%:Q%=RC
% ELSE RC%=TRUE
4170 UNTIL Q% OR INKEY(q%)
4180 ENDPROC
4190 IF INKEY(c1%) THEN PROCchoice(C%,1
):PRINTTAB(0,C%);:TIME=0:REPEAT UNTIL TI
ME>16
4200 IF INKEY(cr%) THEN PROCchoice(C%,
-1):PRINTTAB(0,C%);:TIME=0:REPEAT UNTIL T
IME>16
4210 RETURN
4220 :
5000 DEF FNnode
5010 LOCAL B%
5020 FOR I%=D%+8 TO D%+4 STEP -1
5030 B%=2*B%+POINT(X%+x%(I%MOD6),Y%+y%(
I%MOD6))
5040 NEXT
5050 =B%
5060 :
6000 DEF FNoption
6010 LOCAL Q%,A$
6020 PROCshow:*FX15,1
6030 REPEAT:PRINTTAB(1,R%);
6040 A$=FNupper(GET$)
6050 ON INSTR(c$+l$+r$+u$+d$+"CPRHLSWF"
,A$) GOSUB 6080,6100,6110,6120,6130,6140
,6150,6160,6170,6180,6190,6200,6200 ELSE
VDU 7
6060 UNTIL Q%
6070 =A$
6080 A$=FNalter:IF INSTR("WF",A$) THEN
Q%=TRUE ELSE PROCshow
6090 RETURN
6100 PROCchoice(R%,1):RETURN
6110 PROCchoice(R%,-1):RETURN
6120 R%=(R%+30)MOD31:RETURN
6130 R%=(R%+1)MOD31:RETURN
6140 P%=FALSE:RETURN
6150 P%=TRUE:RETURN
6160 PROCreset:PROCshow:RETURN
6170 FOR I%=0 TO 31:h?I%=c?I%:NEXT:RETU

```

```

RN
6180 PROCload:RETURN
6190 PROCsave:RETURN
6200 Q%=TRUE:RETURN
6210 :
7000 DEF PROCshow
7010 @%=5:CLS
7020 FOR I%=0 TO 30
7030 PRINTTAB(0,I%);:IF u?I% THEN PRINT
"*"; ELSE PRINT" ";
7040 PRINT CHR$(224+2^c?I%);CHR$(ASC("0
")+c?I%);CHR$(224+I%);I%;
7050 NEXT
7060 PRINTTAB(0,31);:IF u?31 THEN PRINT
"*"; ELSE PRINT" ";
7070 PRINT T%;
7080 ENDPROC
7090 :
8000 DEF PROCchoice(R%,A%)
8010 LOCAL C%:C%=c?R%
8020 REPEAT
8030 C%=(C%+A%+5)MOD5
8040 UNTIL (2^C% AND R%)=0
8050 c?R%=C%
8060 PRINTTAB(1,R%)CHR$(224+2^C%);CHR$(
ASC("0")+C%);
8070 ENDPROC
8080 :
9000 DEF FNalter
9010 REPEAT
9020 @%=4:CLS
9030 PRINT"S:""D:""X:""Y:""N:""M:""
9040 PRINTTAB(2,0)S%;
9050 PRINTTAB(2,1)D%;
9060 PRINTTAB(2,2)X%;
9070 PRINTTAB(2,3)Y%;
9080 PRINTTAB(2,4)N%;
9090 PRINTTAB(2,5)M%;
9100 REPEAT
9110 PRINTTAB(5,R2%);
9120 A$=FNupper(GET$)
9130 ON INSTR(l$+r$+u$+d$+"CPZELSWF"+c$
,A$) GOSUB 9170,9180,9190,9200,9210,9220
,9230,9240,9250,9260,9270,9270,9270 ELSE
VDU 7
9140 UNTIL INSTR(c$+"ZELSWF",A$)
9150 UNTIL INSTR(c$+"WF",A$)
9160 =A$
9170 PROCchange(R2%,1):RETURN
9180 PROCchange(R2%,-1):RETURN
9190 R2%=(R2%+5)MOD6:RETURN
9200 R2%=(R2%+1)MOD6:RETURN
9210 P%=FALSE:RETURN
9220 P%=TRUE:RETURN
9230 D%=0:X%=0:Y%=0:RETURN
9240 N%=0:M%=0:P%=FALSE:RETURN
9250 PROCload:RETURN
9260 PROCsave:RETURN

```

```

9270 RETURN
9280 :
10000 DEF PROCchange (R%,A%)
10010 LOCAL MF%,SF%
10020 MF%=1:IF R%=5 SF%=10 ELSE SF%=2
10030 IF INKEY(shift%) THEN MF%=MF%*SF%
10040 IF INKEY(ctrl%) THEN MF%=MF%*SF%^2
10050 ON R%+1 GOSUB 10070,10120,10130,10
150,10170,10180
10060 ENDPROC
10070 O%=S%:S%=(S%+A%+ms%-1)MODms%+1
10080 FOR I%=0 TO 5
10090 x%(I%)=(x%(I%)DIVO%)*S%:y%(I%)=(y%
(I%)DIVO%)*S%
10100 NEXT
10110 PRINTTAB(2,R%)S%:RETURN
10120 D%=(D%+A%+6)MOD6:PRINTTAB(2,R%)D%:
:RETURN
10130 X%=X%+2*A%*MF%:IF ABS(X%)>mx% THEN
X%=-SGN(X%)*mx%
10140 PRINTTAB(2,R%)X%:RETURN
10150 Y%=Y%+4*A%*MF%:IF ABS(Y%)>my% THEN
Y%=-SGN(Y%)*my%
10160 PRINTTAB(2,R%)Y%:RETURN
10170 N%=(N%+A%+31)MOD31:PRINTTAB(2,R%)N
%:RETURN
10180 M%=(M%+A%*MF%+10000)MOD10000:PRINT
TAB(2,R%)M%:RETURN
10190 :
11000 DEF PROCload
11010 LOCAL F%
11020 CLS
11030 INPUT"Load:":"'File?"'F$
11040 IF F$="" THEN PROCshow:ENDPROC
11050 F%=OPENIN(F$)
11060 INPUT#F%,S%,D%,X%,Y%,T%
11070 FOR I%=0 TO 5
11080 INPUT#F%,x%(I%),y%(I%)
11090 NEXT
11100 FOR I%=0 TO 31
11110 c?I%=BGET#F%:u?I%=BGET#F%
11120 NEXT
11130 FOR I%=&3000 TO &7FFF
11140 ?I%=BGET#F%
11150 NEXT
11160 CLOSE#F%
11170 ENDPROC
11180 :
12000 DEF PROCsave
12010 LOCAL F%:CLS
12020 INPUT"Save:":"'File?"'F$
12030 PROCshow
12040 IF F$="" THEN ENDPROC
12050 F%=OPENOUT(F$)
12060 PRINT#F%,S%,D%,X%,Y%,T%
12070 FOR I%=0 TO 5
12080 PRINT#F%,x%(I%),y%(I%)
12090 NEXT

```

```

12100 FOR I%=0 TO 31
12110 BPUT#F%,c?I%:BPUT#F%,u?I%
12120 NEXT
12130 FOR I%=&3000 TO &7FFF
12140 BPUT#F%,?I%
12150 NEXT
12160 CLOSE#F%
12170 ENDPROC
12180 :
13000 DEF FNupper(A$)
13010 IF A$>="a" A$=CHR$(ASC(A$)AND&DF')
13020 =A$
13030 :
14000 DEF PROCchar
14010 DIM C%(31,7)
14020 FOR I%=0 TO 31
14030 C%(I%,4)=8:C%(I%,5)=8:C%(I%,6)=8
14040 IF I% AND 1 THEN C%(I%,4)=C%(I%,4)
OR 2:C%(I%,5)=C%(I%,5) OR 1
14050 IF I% AND 2 THEN C%(I%,1)=C%(I%,1)
OR 1:C%(I%,2)=C%(I%,2) OR 2
14060 IF I% AND 4 THEN C%(I%,0)=C%(I%,0)
OR 8:C%(I%,1)=C%(I%,1) OR 8:C%(I%,2)=C%(
I%,2) OR 8
14070 IF I% AND 8 THEN C%(I%,1)=C%(I%,1)
OR 64:C%(I%,2)=C%(I%,2) OR 32
14080 IF I% AND 16 THEN C%(I%,4)=C%(I%,4
) OR 32:C%(I%,5)=C%(I%,5) OR 64
14090 NEXT
14100 FOR I%=0 TO 31:VDU 23,224+I%
14110 FOR J%=0 TO 7:VDU C%(I%,J%)
14120 NEXT:NEXT
14130 ENDPROC
14140 :
15000 DEF PROCreset
15010 FOR I%=0 TO 31:c?I%=h?I%:NEXT
15020 ENDPROC
15030 :
16000 DEF PROCerror
16010 CLOSE#0:@%=&90A:VDU14
16020 REPORT:PRINT" (Error ";ERR;") at 1
ine ";ERI!"Continue (Y/N/*)?":*FX15,1
16030 REPEAT:A$=GET$
16040 UNTIL INSTR("YyNn*",A$)
16050 IF INSTR("Yy",A$) THEN ENDPROC
16060 IF A$="" THEN INPUT LINE"'"A$:OS
CLI(A$):PRINT'"Press any key":A$=GET$:E
NDPROC
16070 PRINT:*FX4,0
16080 END
16090 :
17000 DATA ms%,0,0,0
17010 DATA dx%,0,dx%/2,dy%,-dx%/2,dy%,-d
x%,0,-dx%/2,-dy%,dx%/2,-dy%
17020 DATA 0,1,0,2,0,1,0,3
17030 DATA 0,1,0,2,0,1,0,4
17040 DATA 0,1,0,2,0,1,0,3
17050 DATA 0,1,0,2,0,1,0,0

```

B

THE BEEBUG MINI WIMP

Part 3

In the third and final part of his series on the BEEBUG MiniWimp, David James shows how this may be used to implement pull-down menus.

Anybody who has used window-based programs on computers such as the Atari ST, the Apple Macintosh or indeed the Archimedes will be familiar with the concept of pull-down menus. At the top of the screen there is a *menu bar* which displays a number of headings. For example, in a page layout program, these might include: 'File', 'Font', and 'Style'.

By moving a pointer onto one of these headings, a menu is displayed. This menu appears to be pulled down from the heading, hence the term 'Pull-down Menus'. Once the menu is on the screen, the pointer can be moved down to the appropriate option. Clicking on that option will then perform the appropriate action, and remove the menu from the screen.

To prevent confusion, most packages that use pull-down menus keep the menu contents unchanged. Thus pulling down, say, the 'File' menu at one stage of the program's execution, does not produce a different result to pulling down the same menu at a different time. This does mean, however, that some options might not be appropriate at all times. For example, saving text from a word processor is not very practical if no text has been entered. To guide the user as to which options are applicable, any option which cannot be performed at the time that a menu is pulled down is displayed in grey text. In this way, the menus themselves do not change during execution, but the choice of options does.

It should be fairly easy to see how to implement pull-down menus using the commands

provided by the BEEBUG MiniWimp system. Because the MiniWimp only returns the pointer position when the user clicks on a selection, we have to insist that a menu is only pulled down when its heading is clicked on. Further, we will also give each menu a *close box* that can be clicked on to remove the menu without selecting any choices.

Listing 1 is an example program that shows how pull-down menus can be implemented. The main part of the program is between lines 100 and 260. The rest of the program is a set of functions and procedures to implement the pull-down menus. You should enter the program and save it. Before running the program, the MiniWimp ROM from BEEBUG Vol.7 No.1 must be installed into sideways RAM. Instructions for doing this were given in the original article. When the program is run, three menu headers appear, these being 'Sounds', 'Colours', and 'Finish'. By moving the pointer to 'Sounds' or 'Colours' with the cursor keys, and clicking using the Copy key, the appropriate menu can be pulled down. The 'Sounds' menu offers a choice of two sounds, whereas the 'Colours' menu allows the background colour to be changed. You can choose a particular option by pointing at it and clicking. If you click on the small box containing the cross, the menu is removed without any other action being taken. Clicking on 'Finish' will quit the program.

To use pull-down menus in your own programs, all you have to do is to incorporate a number of standard procedures from listing 1 as detailed below, and ensure that the MiniWimp code itself has been loaded into sideways RAM.

PROChader(header\$)

This takes as its argument a string that consists of the menu headers, separated by spaces. The routine draws a white box with rounded corners across the top of the screen, and prints the menu headers inside this.

FNarea(left,bottom,right,top)

This checks the current position of the pointer against the box whose co-ordinates are given in the function call, and returns the value TRUE if the pointer is inside the box. Otherwise, the

value FALSE is returned. This function is used to test if the pointer has been clicked on any of the menu headers.



FNmenu(options,flags,X,Y,width)

This is the main function for displaying and handling menus. The first parameter is a string consisting of the various choices, each separated by an '*'. The second parameter, which should be an integer, determines whether a particular option is greyed out or not. This value should be thought of as a binary number, with one bit per menu option. Bit 0 corresponds to the first menu choice, bit 1 to the second, and so on. If the bit is clear, then that choice is allowable, but if the bit is set, the choice is printed in grey, and cannot be selected. The third and fourth parameters are the character co-ordinates of the top-left corner where the menu should appear, and the final parameter is the width of the menu in characters. FNmenu exits when the user clicks on an option, and will return either the number of that option (starting with zero at the top), or -1 if the close box was clicked on.

PROCprint is a routine to print characters proportionally spaced. In other words, no large gaps are left between letters. This routine is called by FNmenu, and must be included in your own programs.

PROCassemble generates some machine code that fills the screen with a stippled background colour. This is activated using CALL &900.

By following the example program, and the instructions for using the MiniWimp commands given in BEEBUG Vol.7 No.1, it

should be fairly easy to incorporate pull-down menus in your own programs. There are, of course, many other applications for the BEEBUG MiniWimp, particularly if combined with the Icon Designer (BEEBUG Vol.7 No.2). Altogether, this provides a complete WIMP environment for your BBC micro.

Listing 1

```

10 REM Program Pull Down Menus
20 REM Version B 1.0
30 REM Author David James
40 REM BEEBUG Aug/Sept 1988
50 REM Program subject to copyright
60 :
100 MODE 4
110 ON ERROR IF ERR=17 THEN 260 ELSE R
EXPORT:END
120 VDU 23;8202;0;0;0;
130 PROCassemble
140 *MWSETUP
150 FS%=0
160 COLOUR 129:COLOUR 0
170 CLS
180 CALL &900
190 PROCheader("Sounds Colours Finish"

200 REPEAT end=FALSE
210 *MWPOINTER
220 IF FNarea(3,1,8,0) PROCsounds
230 IF FNarea(10,1,16,0) PROCcols
240 IF FNarea(18,1,23,0) end=TRUE
250 UNTIL end
260 MODE 7:END
270 :
1000 DEF PROCassemble
1010 FOR I%=0 TO 3 STEP 3
1020 P%=&900:[OPT I%
1030 LDA#0:STA&73:LDA#&58:STA&74
1040 .b1 LDY#0:.b2 LDA block%,Y
1050 STA(&73),Y:INY:CPY#8:BNEb2
1060 TYA:CLC:ADC&73:STA &73
1070 LDA#0:ADC&74:STA&74
1080 CMP#&80:BNEb1:RTS
1090 .block% EQU&B57EE7DA
1100 EQU&5BE77EAD
1110 ]NEXT:ENDPROC
1120 :
1130 DEF PROCsounds
1140 ch%=FNmenu("Sound 1*Sound 2",FS%,2
,1,7)
1150 IF ch%=-1 THEN GOTO 1180
1160 SOUND 1,-15,100+100*ch%,10
1170 FS%=FS% OR 2^ch%
```

```

1180 *MWSHUT
1190 ENDPROC
1200 :
1210 DEF PROCcols
1220 ch%=FNmenu("Red*Green*Yellow*Blue*
Magenta*Cyan*White",0,10,1,7)
1230 IF ch%=-1 THEN GOTO 1250
1240 VDU 19,1,ch%+1,0;
1250 *MWSHUT
1260 ENDPROC
1270 :
1280 DEF PROCheader(H$)
1290 GCOL 0,3:MOVE 0,981:MOVE 0,1023
1300 PLOT 85,1279,981:PLOT 85,1279,1023
1310 GCOL 0,0:MOVE 0,977:DRAW 1279,977
1320 REM define & print rounded corners
1330 VDU23,128,248,224,192,128,128,0,0,
0,23,129,31,7,3,1,1,0,0,0
1340 PRINTCHR$128TAB(39)CHR$129
1350 VDU5:MOVE 96,1015:PRINTH$:VDU4
1360 ENDPROC
1370 :
1380 DEF FNarea(X1%,Y1%,X2%,Y2%)
1390 =(X%>=X1% AND X%<=X2% AND Y%<=Y1%
AND Y%>=Y2%)
1400 :
1410 DEF FNmenu(opt$,F%,XC%,YC%,XL%)
1420 VDU23,130,0,255,0,255,0,0,0,0,23,1
31,193,162,148,136,148,162,193,255
1430 N%=0:FOR L%=1 TO LENopt$
1440 IF MID$(opt$,L%,1)="*"THEN N%=N%+1
1450 NEXT L$:nopt%=N%

```

```

1460 OSCLI"MWOPEN "+STR$XC%+" "+STR$(YC
%+nopt%+2)+" "+STR$(XC%+XL%)+ " "+STR$YC%
1470 PRINTSTRINGS(XL%,CHR$130);CHR$131
1480 FOR O%=0 TO nopt%
1490 pos%=INSTR(opt$,"*")
1500 option$=LEFT$(opt$,pos%-1)
1510 opt$=RIGHT$(opt$,LEN(opt$)-pos%)
1520 PROCprint(option$,O%)
1530 NEXT O%
1540 REPEAT
1550 *MWPOINTER
1560 choice%=Y%-YC%-1
1570 UNTIL choice%>=-1 AND choice%<nopt
%+1 AND ((F% AND 2^choice%)=0)
1580 =choice%
1590 :
1600 DEF PROCprint(M$,O%)
1610 VDU 5:VDU 29,(XC%+1)*32;1020-((YC%
+O%+1)*32);
1620 mw%=0
1630 FOR M%=1 TO LENM$
1640 L$=MID$(M$,M%,1)
1650 ?&980=ASCL$:X%=&80:Y%=9
1660 A%=10:CALL &FFFL
1670 IF (F% AND 2^O%)<>0 !&981=!&981 AN
D &55AA55AA:!&985=!&985 AND &55AA55AA
1680 VDU23,128,!&981;!&983;!&985;!&987;
1690 MOVE28*(M%-1)+mw%,0:VDU128
1700 IF INSTR("MWmw",L$) mw%=mw%+4
1710 NEXT
1720 VDU 4,29,0,0;
1730 ENDPROC

```

B

PAINTBOX AND ILLUSTRATOR (Continued from page 7)

characters. The display can be adjusted by mouse and keyboard before the dump begins. A separate utility also allows setting of some non-Epson printer defaults, which are automatically stored on disc and adopted thereafter. And a word processor can be used independently to print text around an image.

The program saves its screen dumps as standard mode 4 screen files, which could work with other software. There are drawbacks to this, and it is not mentioned in the manual, so I should not criticise it, but it could be a valuable extra. The cost is in the disc space which is wasted every time you save 10k of screen in order to store only 60% of it.

Some graphics screens and alternative brush and pattern menus are provided on disc. This does not constitute an 'extensive library' as the manual says, and they don't justify further comment.

Illustrator is well priced at fifty pounds and suffers most from its exaggerated claims. It is not one of those excellent, small, user-friendly packages which do little but do it admirably. In some areas it attempts a little too much and fails. However, most of it is thoroughly useable, it creates and dumps fairly large mode 4 images to disc and to the printer and, by using standard screen files, it can integrate with Paintbox and other software.

B

sacrificed. Thus in the case of the program for French, pressing the @-key will produce the letter 'a' with a circumflex accent over it. The end of the statement is a comment indicating the character which will be printed. Thus a library of character definitions can be built up and DATA statements can be inserted and deleted as required. Care must be taken, though, not to redefine a character twice, since only the second definition will have any effect. It is useful to make a diagram of the keyboard, indicating on the keys the characters which will be printed. The last DATA statement must always be a double asterisk.

There are two program listings. The first is the menu program which automatically resets PAGE and calls the appropriate file, whose name must be the same as that of the language being used. You can use *OPT4,3 to ensure that the menu program will run automatically when you press Shift-Break. The second listing is the character redefinition program itself and the DATA statements for the French language.

We have also listed here the DATA statements for German. Simply replace the DATA statements in the program listing as required. The magazine disc/tape contains the character definitions for the following languages: German, Spanish, Turkish and Greek (as well as French). Some readers may require slightly different letters, for Scandinavian languages, for example, or else you may wish to insert mathematical symbols in your text. Details of how to make up 'do-it-yourself' characters will be included in a subsequent article.

NOTE: Most printers require a DIP switch to be set to allow internal RAM to be used for redefining characters. The alternative switching usually allocates this RAM as a printer buffer. Check with your printer's manual. On an Epson FX80, pin 4 of DIP switch No.1 should be set off.

```
10 REM Program ViewF
20 REM Version B1.0
30 REM Author Eddy Hunt
40 REM BEEBUG Aug/Sept 1988
50 REM Program subject to copyright
60 :
100 MODE 7:ON ERROR GOTO 230
```

```
110 r$=CHR$129:y$=CHR$131:g$=CHR$130
120 PROCdhy("Multi-Language Word Processsing")
130 PRINT'"Word Processor:";y$;"View"
140 PRINT'"Printer Required:";y$;"Epson FX80"
150 PRINT'"Printer on-line? (Y/N)";
160 REPEAT:G=GET AND &DF:UNTIL G=78 OR G=89
170 X%=(G=89)
180 PROCdhy("View - Foreign Character Sets")
190 PROCmenu
200 PROCkeydefs
210 END
220 :
230 REPORT:PRINT" at line ";ERL:END
240 :
1000 DEF PROCmenu
1010 PRINT'
1020 PRINT'r$;"f0";g$;"French"
1030 PRINT'r$;"f1";g$;"German"
1040 PRINT'r$;"f2";g$;"Spanish"
1050 PRINT'r$;"f3";g$;"Turkish"
1060 PRINT'r$;"f4";g$;"Greek"
1070 PRINT'
1080 ENDPROC
1090 :
1100 DEF PROCkeydefs
1110 *KEY 0 MODE 3|MPAGE=PAGE+&600|MCH.
"FRENCH"|M*WORD|M
1120 *KEY 1 MODE 3|MPAGE=PAGE+&600|MCH.
"GERMAN"|M*WORD|M
1130 *KEY 2 MODE 3|MPAGE=PAGE+&600|MCH.
"SPANISH"|M*WORD|M
1140 *KEY 3 MODE 3|MPAGE=PAGE+&600|MCH.
"TURKISH"|M*WORD|M
1150 *KEY 4 MODE 3|M PAGE=PAGE+&600|MCH
"GREEK"|M*WORD|M
1160 ENDPROC
1170 :
1180 DEF PROCdhy(a$)
1190 x%=18-LEN(a$) DIV 2
1200 CLS:PRINTTAB(x%,1)CHR$141;y$;a$:PRINTTAB(x%,2)CHR$141;y$;a$
1210 ENDPROC
```

* * * * *

```
10 REM Program French
20 REM version B1.0
30 REM Author Eddy Hunt
40 REM BEEBUG Aug/Sept 1988
50 REM Program subject to copyright
60 :
100 ON ERROR GOTO 200
```

```

110 *FX20,6
120 IF X% PROCinitprint
130 REPEAT
140 READ cd$
150 IF cd$<>"*" THEN PROCchargin(cd$)
160 UNTIL cd$="*"
170 IF X% PROCset
180 END
190 :
200 REPORT:PRINT" at line ";ERL:END
210 :
1000 DEF PROCchargin(cd$)
1010 rc%=ASC(cd$)
1020 IF X% THEN PROCdefprint(rc%,cd$)
1030 PROCdefchar(rc%,cd$)
1040 ENDPROC
1050 :
1060 DEF PROCdefprint(rc%,cd$)
1070 LOCAL i%,p%:p%=3
1080 ND%=MID$(cd$,2,1)="^"
1090 VDU2
1100 VDU1,27,1,38,1,0,1,rc%,1,rc%,1,11-
128*ND%
1110 FOR i%=0 TO 8
1120 VDU1,EVAL("&"+MID$(cd$,p%+2*i%,2))
1130 NEXT
1140 VDU1,0,1,0
1150 VDU3
1160 ENDPROC
1170 :
1180 DEF PROCdefchar(rc%,cd$)
1190 LOCAL i%,p%:p%=22
1200 VDU23,rc%
1210 FOR i%=0 TO 7
1220 VDU EVAL("&"+MID$(cd$,p%+2*i%,2))
1230 NEXT
1240 ENDPROC
1250 :
1260 DEF PROCinitprint
1270 VDU2
1280 VDU1,27,1,64
1290 VDU1,27,1,58,1,0,1,0,1,0
1300 VDU3
1310 ENDPROC
1320 :
1330 DEF PROCset
1340 VDU2
1350 VDU1,27,1,37,1,1,1,0
1360 VDU1,27,1,ASC("M")
1370 VDU3
1380 ENDPROC
1390 :
9000 DATA ";^040A208A600A201C02-10083C0
63E663E00-a-grave"

```

```

9010 DATA "@^020550159005500E01-1824003
C063E663E-a-circumflex"
9020 DATA "[^1C22082248A2082210-08103C6
67E603C00-e-acute"
9030 DATA "J^1C2208A24822082210-10083C6
67E603C00-e-grave"
9040 DATA "^^0E1144118411441108-1824003
C667E603C-e-circumflex"
9050 DATA "_d384401440146004400-003C666
0663C1838-c-cedilla"
9060 DATA "|^00A2003E0082020000-6600381
818183C00-i-dotdot"
9070 DATA "jd00006D006E00000000-0000181
800181830-semicolon"
9080 DATA "~^000000006090009060-0C12120
C00000000-degree"
9090 DATA "^3E002A40AA002A0022-08107E6
07C607C00-E-acute"
9100 DATA **

```

* * * * *

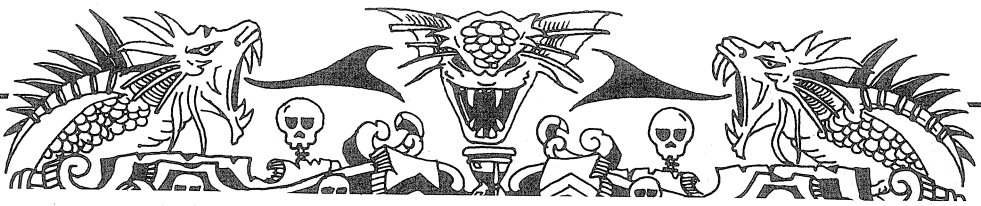
German

```

9000DATA "j^040A208A600A201C02-10083C0
63E663E00-a-grave"
9010DATA "|^020550159005500E01-1824003
C063E663E-a-circumflex"
9020DATA "^^1C22082248A2082210-08103C6
67E603C00-e-acute"
9030DATA "^408000800A80108060-3C660C1
818001800-?"
9040DATA "~^0E1144118411441108-1824003
C667E603C-e-circumflex"
9050DATA "?^068814204420148806-42183C6
67E666666-A-umlaut"
9060DATA ">^1CA20022002200A21C-423C666
66666663C-O-umlaut"
9070DATA "+^3C028002000280023C-6600666
66666663C-U-umlaut"
9080DATA "/^040AA00A200AA01C02-42003C0
63E663E00-a-umlaut"
9090DATA "\^000E51009100510E00-18243C6
666663C00-o-circomflex"
9100DATA ".^001CA2002200A21C00-42003C6
666663C00-o-umlaut"
9110DATA ";^003C80020002803C02-6600006
666663E00-u-umlaut"
9120DATA "@^7E8000801280126C00-003C667
C667C6060-double-S"
9130DATA "{^00A2003E0082020000-6600381
818183C00-i-dotdot"
9140DATA "<^000006000600000000-0000000
000001818-fullstop"
9150DATA "*d00006D006E00000000-0000181
800181830-semicolon"
9160DATA **

```

B



ADVENTURE GAMES by Mitch ADVENTURE GAMES

How would you like a fist in your ear? Sounds painful doesn't it? Well in fact it can be fun when FIST stands for Fantasy Interactive Scenarios by Telephone. You are no doubt familiar with the Multi-User Dungeon (MUD) games which may be played by connecting your micro, via a modem and a telephone line, to a distant computer. The problem with that system is the initial cost of the modem plus the associated software. The idea behind FIST is that you may play the game instantly using only a push-button telephone (no modem or computer of your own is required). By dialling (0898) 800876 you will be transported in sound to the courtyard of Castle Mammon.

With the final click of the connection made, reality melts away and over the line come the hollow sounds from the other world. While a cold evening wind moans around the battlements, a ghostly voice whispers a description of your surroundings into your ear. From nearby comes the sudden scream of a maiden falling into a pit which has opened up in the middle of the cobbled yard. Now you must make the first of many decisions which will lead you down into the monster-infested dungeons. The ghostly narrator will inform you of the possible actions you may take, and it is up to you to make your decision by pressing an appropriate key on your telephone key pad.

The game was devised by Steve Jackson, who is the author of a series of Fighting Fantasy books, and FIST is such a novel brought to life with the aid of the Voicetek telecommunication equipment. The system accesses a Winchester disc holding 150 Mbytes of digitised sounds, from which it instantly finds and plays the appropriate section of data to let you hear the consequence of your choice. Should you decide to rescue the maiden, rather than 'leg it' for the nearest bolt-hole, you will be rewarded by a whistling blow which will send you crashing and clanking down into the darkness. Prepare your sword arm quickly, for here in the gloom a bellowing creature is limbering up in preparation for separating your chivalrous head from your shoulders.

When the system went live on March 1st the lines were jammed with thousands of adventurers

anxious to be the first to find gold in the castle's depths. Finding gold is the measure of success in this game and it is also a way in which you can reduce your costs. Each month the players who have amassed the most gold are awarded prizes, which range from T-shirts and rings, up to 'real' money. Eerie music, snorting monsters and death rattles pour down the line as you edge deeper into the dungeon. There are approximately 50 different rooms to visit, and just as many monsters. One room contains The Adjuster, who behaves like a TV game-show host. Here you will have the option to 'Come on Down' and gamble all or part of your gold answering the Adjuster's questions in an effort to increase your score.

When you finally emerge from the horrors of the castle's dungeons you will then see the most frightening apparition of all - the telephone bill! 'Aye there's the rub'. At a staggering 38p a minute peak-time and 25p off-peak, you will need to be already in possession of a sack of gold before you go treasure-seeking in Castle Mammon. The company accept that the cost is expensive, but unfortunately it is British Telecom who sets the rates and not them. A new subscription scheme which could cut costs considerably is being considered, but this is not yet finalised.

An Adventurer's Guild has been set up to provide additional services to users and they also publish a quarterly newsletter. Whilst it is possible to play the game using the rotary-dial telephones, the Guild sell a Tone Dialer for approximately £10. This device connects to the mouthpiece of your handset and enables you to access some of the more sophisticated options. You may contact the Guild at: *Rex House, 4-12 Lower Regent Street, London, SW1Y 4PE*

You may also get more information on FIST by sending a large S.A.E. to *Computerdial, 6 Leaplace Road, Guilford GU1 4JU*.

The game sounds like an exciting radio play, and when you couple this with the imagination of an adventure enthusiast, the resulting mixture is sure to please. Cost is of course the only reason for not instantly throwing yourself onto the nearest phone, but if used sparingly, FIST will bring a new dimension to your hobby.

B

Matrices

In Basic

(Part 2)

Jan Stuurman adds some powerful new functions to his Basic extensions to provide a complete matrix manipulation package in BBC Basic.

The program with last month's article provided some simple operations that could be performed on matrices stored as Basic arrays. This month's program extends these operations by adding two very powerful matrix functions. The first of these is a command to calculate the inverse of a square matrix, while the second builds on this to give a command that will solve a set of simultaneous equations.

Listing 1 is a series of lines to add to the program from last month. First of all, load in last month's program, and ensure that the line numbering is as in the original article. Then, enter the additional lines from listing 1. The first two of these overwrite existing lines. Users of Basic I should replace the last two lines of listing 1 with:

```
10181 stfat2=&A36E:ptmt2=&A7F3
10182 divmfa=&A6AD
```

You should, of course, have included the Basic I modifications from last month as well. Once the new lines have been entered, the program should be saved, using a different name from the original just in case something goes wrong. When the new program is run, it will save the matrix handling machine code to disc under the name 'MAT', as before. To install the routines, type *MAT from within Basic. The new code loads in at &7500, and so mode 7, or any shadow mode, should be selected first, and HIMEM set to &7500.

MATRIX INVERSION

The inverse of a square matrix is another square matrix of the same size, such that when this is multiplied by the original matrix, the result is the *identity* matrix. You may remember from last month that the identity matrix is a square

matrix that contains all zeros, except for the leading diagonal, which contains all ones.

As an example, consider the two by two matrix A, which contains the elements:

$$\begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}$$

The inverse of this matrix is:

$$\begin{pmatrix} -2.5 & 1.5 \\ 2 & -1 \end{pmatrix}$$

because, multiplying the two together, you get:

$$\begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix} * \begin{pmatrix} -2.5 & 1.5 \\ 2 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

which is the two by two identity matrix. The process of finding the inverse of a matrix can be very time consuming, especially for large matrices. However, using the Basic extensions, the inverse of a matrix is easily calculated with the command:

MAT A = INV(B)

which will take the square matrix in array B, invert it, and put the result in array A. Clearly, A and B must have the same number of dimensions, and because the inverse of a non-square matrix is un-defined, both A and B must have an equal number of rows and columns. The code gives a 'Bad DIM' error if this is not the case.

There is one further problem that can occur when trying to evaluate the inverse of a matrix. This is simply that some matrices do not have an inverse. Such matrices are said to be 'singular', and an example is the two by two matrix:

$$\begin{pmatrix} 2 & 1 \\ 4 & 2 \end{pmatrix}$$

The INV operation will generate an error message if its argument is a singular matrix.

A final word of warning. To save workspace, the INV function corrupts the value of its argument. So,

MAT A = INV(B)

will set A to the inverse of B, but will also corrupt B. If you do not want this to happen, you should explicitly save matrix B first, and restore it afterwards. This can be done using:

MAT C = COP(B)

MAT A = INV(B)

MAT B = COP(C)

after having defined C to be an array of the same size as A and B.

SIMULTANEOUS EQUATIONS

In mathematics, you often encounter problems such as:

Given that:

$$4x + 3y = 12 \text{ and}$$

$$2x + y = 7$$

what are the values of 'x' and 'y'?

Problems such as this are known as simultaneous equations, because you must find solutions for all the separate equations simultaneously. This example only has two unknown variables, but in general, the equations can have any number of unknowns. The main proviso is that to find a solution of a set of simultaneous equations with 'n' unknowns requires 'n' separate equations. Further, all these equations must be 'linearly independent', which means that no equation can be a simple multiple of another one. As an example of this, consider the equations:

$$4x + 3y + 2z = 12$$

$$3x + 7y - 5z = 15$$

$$8x + 6y + 4z = 24$$

There may appear to be three different equations here, but if you study them, you will find that the last one is simply the first one doubled. This means that, given these three equations, you could not find unique values for 'x', 'y' and 'z'.

One way you might try and solve simultaneous equations is by guessing at the values. This can sometimes work with two unknowns, but try it for twenty! A much better solution comes when you represent the equations as a matrix multiplication sum. For example, the first set of equations given above could be written as:

$$\begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 12 \\ 7 \end{pmatrix}$$

Try multiplying this out if you are not sure that it is the same. To save writing the matrices each time, we will refer to the square matrix as 'M', the unknown matrix as 'X', and the result matrix as 'R'. This gives us:

$$M * X = R$$

Now, suppose we find the inverse of 'M', which is written as 'M⁻¹'. We can now multiply both sides of our matrix equation by 'M⁻¹'. This gives:

$$M^{-1} * M * X = M^{-1} * R$$

However, we know that if you multiply a matrix by its inverse you get the identity matrix, (that is the definition of the inverse), so the sum reduces to:

$$I * X = M^{-1} * R$$

But, multiplying a matrix by the identity, leaves that matrix unchanged. So, we end up with:

$$X = M^{-1} * R$$

In other words, we can find the solutions to a set of simultaneous equations by multiplying the inverse of the matrix made up from the equations by the result matrix (R) of the equations.

To prove this, the inverse of 'M' in the above example is:

$$\begin{pmatrix} -0.5 & 1.5 \\ 1 & -2 \end{pmatrix}$$

Multiplying 'R' by this gives:

$$\begin{pmatrix} -0.5 & 1.5 \\ 1 & -2 \end{pmatrix} * \begin{pmatrix} 12 \\ 7 \end{pmatrix} = \begin{pmatrix} 4.5 \\ 2 \end{pmatrix}$$

showing that x=4.5, and y=-2. You can try these solutions out for yourself if you want.

While you could use the operations already provided by the MAT statement to solve simultaneous equations, there is a special operation to perform everything in one go. This is the comma operator, and is written as:

$$MAT A = B , C$$

which performs the operation A=INV(B)*C. A is the matrix for the result, B is the square matrix made up from the equations, and C is the result matrix of the equations. So, the above problem could have been solved with the commands:

```
DIM A(1,1),B(0,1),C(0,1)
A(0,0) = 4 : A(1,0) = 3
A(0,1) = 2 : A(1,1) = 1
B(0,0) = 12 : B(0,1) = 7
MAT C = A , B
x = C(0,0) : y = C(0,1)
```

MASTER AND COMPACT BASIC

As stated last month, the matrix machine code as it stands will not work on a Master 128 or a Compact. While it is relatively easy to adapt the program to run with Basic I, because all the routines used in Basic II are also in Basic I, the situation is more complex with later versions of Basic. This is because some of the equivalent routines in Basic IV and Basic VI are not only in

different locations, but have been significantly modified in the way they behave.

To create a Master or Compact version of the matrix program, first of all enter the standard Basic II version, by typing in listing 1 from last month, but omitting lines 10080 to 10180. Then add listing 1 from this month. Next, all occurrences of '&4B' in the program must be changed to read '&4A'. There are twenty-three of these to change, the lines in question being:

2020,2070 (2), 2190,2380,2410,2660,2900
2920,3210,3400,3780,4410,4420,4430
4560,4720,4750,4860,4870 (2),4880 (2)

(A number in brackets indicates the number of changes to make to that particular line). You should then do the same for all the '&4C's, which should be changed to '&4B'. There are sixteen of these in lines:

2030,2080,2200,2390,2420,2670,2910
2930,3220,3410,3790,4440,4570,4730
4760,4890

Finally, enter either listing 2 if you are using a Master 128, or listing 3 if you have a Compact, and save the complete program. This can now be run as with the other versions.

Listing 1

```
1880 EQUUS ",":EQUW lin
1940 EQUUS "INV":EQUW 0:EQUW inv
3440 .inv JSR chksqr
3450 LDA oper+3:LDX #2:JSR chkar
3460 BNE wrgdim:BCC wrgdim:JSR idn
3470 LDA dim2:STA dim:STA dim+1
3480 JMP jordan
3490 :
3500 .lin LDA oper:LDX #2:JSR chkar
3510 BCC wrgdim
3520 LDA dim2:CMP dim2+1:BNE wrgdim
3530 LDA oper+2:LDX #4:JSR chkar
3540 BNE wrgdim:BCC wrgdim:JSR copy
3550 LDA dim4:STA dim
3560 LDA dim4+1:STA dim+1
3570 JMP jordan
3580 :
3590 .jordan JSR ldfan1:JSR stfat2
3600 LDA dim2+1:STA ldim:CLC
3610 ASL A:ASL A:ADC ldim:STA off
3620 ADC #5:STA doff
3630 LDA base2:STA base4
3640 LDA base2+1:STA base4+1
3650 LDA #0:STA qoff:DEC ldim
3660 .luloop JSR pivot
3670 .lloop1 JSR determ:JSR gauss
3680 CLC:LDA base4:ADC doff:STA base4
```

```
3690 BCC dnoc:INC base4+1:.dnoc
3700 INC qoff:LDA qoff:CMP ldim
3710 BCC luloop:BEQ lloop1
3720 .bsloop DEC qoff
3730 SEC:LDA base4:SBC doff:STA base4
3740 BCS dnob:DEC base4+1:.dnob
3750 JSR bcksub
3760 LDA qoff:CMP #1:BNE bsloop:RTS
3770 :
3780 .determ LDA base4:STA &4B
3790 LDA base4+1:STA &4C:JSR ldfam
3800 BEQ detzer:JSR stfat1
3810 JSR pntmt2:JSR mufamo
3820 JSR stfat2:RTS
3830 :
3840 .detzer BRK:EQUW &64
3850 EQUUS "Determinant zero":EQUW 0
3860 :
3870 .pivot
3880 LDA base4:STA pbase
3890 LDA base4+1:STA pbase+1
3900 LDA qoff:STA poff:STA mdim
3910 JSR stpiv
3920 .check INC mdim
3930 CLC:LDA pbase:ADC #5:STA pbase
3940 BCC cnoc:INC pbase+1:.cnoc
3950 JSR cmpare:BEQ cnext:JSR stpiv
3960 LDA mdim:STA poff:JSR swap
3970 .cnext LDA mdim:CMP ldim:BCC check
3980 RTS
3990 :
4000 .swap LDA &472:EOR #&80:STA &472
4010 LDX #0:JSR mswap:LDX #2
4020 .mswap
4030 CLC:LDA qoff:ASL A:ASL A:ADC qoff
4040 ADC base,X:STA cbase
4050 LDA base+1,X:ADC #0:STA cbase+1
4060 CLC:LDA poff:ASL A:ASL A:ADC poff
4070 ADC base,X:STA qbase
4080 LDA base+1,X:ADC #0:STA qbase+1
4090 LDA dim,X:STA cdim
4100 .movfpn LDY #4
4110 .movbyt LDA (qbase),Y:PHA
4120 LDA (cbase),Y:STA (qbase),Y
4130 PLA:STA (cbase),Y:DEY:BPL movbyt
4140 CLC:LDA cbase:ADC off:STA cbase
4150 BCC mnoc1:INC cbase+1:.mnoc1
4160 CLC:LDA qbase:ADC off:STA qbase
4170 BCC mnoc2:INC qbase+1:.mnoc2
4180 DEC cdim:BNE movfpn:RTS
4190 :
4200 .stpiv LDY #4
4210 .stloop LDA (pbase),Y:STA fpbm,Y
4220 DEY:BPL stloop
4230 LDA &3D:AND #&7F:STA &3D:RTS
4240 :
```

```

4250 .compare LDY #0
4260 LDA (pbase),Y:CMP fpbm:BNE neq
4270 INY:LDA (pbase),Y:AND #&7F
4280 CMP fpbm+1:BNE neq
4290 .cmloop INY
4300 LDA (pbase),Y:CMP fpbm,Y:BNE neq
4310 CPY #4:BCC cmloop
4320 .leq LDA #0:RTS
4330 .neq BCC leq:LDA #1:RTS
4340 :
4350 .gauss
4360 LDX #0:JSR divrow
4370 LDX #2:JSR divrow
4380 LDA qoff:CMP ldim:BEQ greedy
4390 STA mdim
4400 .galoop INC mdim:SEC
4410 LDA mdim:SBC qoff:STA &4B
4420 CLC:ASL A:ASL A:ADC &4B
4430 ADC base4:STA &4B
4440 LDA base4+1:ADC #0:STA &4C
4450 JSR ldfam:JSR stfat1
4460 LDX #0:JSR subrow
4470 LDX #2:JSR subrow
4480 LDA mdim:CMP ldim:BCC galoop
4490 .greedy RTS
4500 :
4510 .divrow CLC:LDA qoff:ASL A:ASL A
4520 ADC qoff:ADC base,X:STA pbase
4530 LDA base+1,X:ADC #0:STA pbase+1
4540 LDA dim,X:STA cdim
4550 .diloop JSR ldfat1
4560 LDA pbase:STA &4B
4570 LDA pbase+1:STA &4C
4580 JSR divmfa:JSR stfam
4590 CLC:LDA pbase:ADC off:STA pbase
4600 BCC dinoc:INC pbase+1:.dinoc
4610 DEC cdim:BNE diloop:RTS
4620 :
4630 .subrow CLC:LDA qoff:ASL A:ASL A
4640 ADC qoff:ADC base,X:STA pbase
4650 LDA base+1,X:ADC #0:STA pbase+1
4660 CLC:LDA mdim
4670 ASL A:ASL A:ADC mdim
4680 ADC base,X:STA cbase
4690 LDA base+1,X:ADC #0:STA cbase+1
4700 LDA dim,X:STA cdim
4710 .suloop
4720 LDA pbase:STA &4B
4730 LDA pbase+1:STA &4C
4740 JSR ldfam:JSR pntmt1:JSR mufamo
4750 LDA cbase:STA &4B
4760 LDA cbase+1:STA &4C
4770 JSR subfam:JSR stfam
4780 CLC:LDA pbase:ADC off:STA pbase
4790 BCC snoc1:INC pbase+1:.snoc1
4800 CLC:LDA cbase:ADC off:STA cbase

```

```

4810 BCC snoc2:INC cbase+1:.snoc2
4820 DEC cdim:BNE suloop:RTS
4830 :
4840 .bcksub LDA qoff:STA mdim
4850 .baloop DEC mdim:SEC
4860 LDA qoff:SBC mdim:STA &4B:CLC
4870 ASL A:ASL A:ADC &4B:STA &4B
4880 SEC:LDA base4:SBC &4B:STA &4B
4890 LDA base4+1:SBC #0:STA &4C
4900 JSR ldfam:JSR stfat1
4910 LDX #0:JSR subrow
4920 LDA mdim:BNE baloop:RTS
10075 fpbm=&3C
10181 stfat2=&A37D:pntmt2=&A7ED
10182 divmfa=&A6AD

```

Listing 2

```

1500 LDA #3:STA &39:JSR schvar
1970 .zer JSR ldfan0:BRA fillar
8000 .pntmt2 LDA #&71:STA &4A
8010 LDA #4:STA &4B:RTS
8020 .stfat2 JSR pntmt2:JMP stfat
10080 cont =&9005:chkend=&9BA6
10090 syntax=&9B69:getcha=&8EE0
10100 schvar=&8087:varnfd=&AD8C
10110 baddim=&952C:ldfan0=&A6B4
10120 ldfan1=&A5D8:stfam =&A519
10130 getnsb=&9D3B:mismat=&9092
10140 citof =&8185:ldfam =&A541
10150 addmfa=&A68D:subfam=&A68A
10160 nmlfa =&81F7:stfat1=&A511
10170 ldfat1=&A539:mufamo=&A6A6
10180 pntmt1=&A592:misbrc=&AD9E
10181 stfat =&A519
10182 divmfa=&A5EE

```

Listing 3

```

1500 LDA #3:STA &39:JSR schvar
1970 .zer JSR ldfan0:BRA fillar
8000 .pntmt2 LDA #&71:STA &4A
8010 LDA #4:STA &4B:RTS
8020 .stfat2 JSR pntmt2:JMP stfat
10080 cont =&8FE6:chkend=&9B86
10090 syntax=&9B49:getcha=&8EB9
10100 schvar=&8087:varnfd=&ACEA
10110 baddim=&950D:ldfan0=&A647
10120 ldfan1=&A53E:stfam =&A469
10130 getnsb=&9D1B:mismat=&9071
10140 citof =&81A9:ldfam =&A49A
10150 addmfa=&A626:subfam=&A620
10160 nmlfa =&821E:stfat1=&A461
10170 ldfat1=&A492:mufamo=&A600
10180 pntmt1=&A4D7:misbrc=&ACFC
10181 stfat =&A469
10182 divmfa=&A550

```


1st COURSE

Just Scrolling

Mike Williams describes a variety of text scrolling routines by Lindsey Cullen.

In a series of three articles under our First Course banner, I described the use of Basic's string handling functions with a variety of simple examples. This month I propose to deal further with string handling by describing a set of fascinating routines for scrolling text written by Lindsey Cullen.

There are many opportunities for spicing up your screen displays, not least by looking for more imaginative ways of placing text on the screen than the mere use of the PRINT instruction. The procedures listed here illustrate a variety of techniques which you may use in your own programs. Even more, I hope they will provoke you to devise even better alternatives.

The first idea we shall look at is that of scrolling text sideways onto the screen. The following procedure, which may be used in any mode, will scroll any given message so that it appears to to 'enter' the screen from the right-hand side. The full text ends up positioned centrally across the screen.

```
100 DEF PROCdisplay1(a$,y%,w%)
110 r%=LEN(a$):s%=(w%-r%)/2
120 a$=a$+CHR$32
130 FOR loop%=0 TO r%+s%
140 ch$=MID$(a$,1,loop%)
150 PRINTTAB(w%-2-loop%,y%);ch$
160 TIME=0:REPEAT UNTIL TIME>8
170 NEXT
180 ENDPROC
```

The message to be displayed is specified as the parameter a\$. The second parameter (in common with all the other procedures) specifies which line on the screen is to be used, and another parameter is used to specify a screen width (20, 40

or 80 columns). If you are only going to use the procedure in a single mode, this parameter and all references to it may be omitted, and replaced by a suitable fixed value.

We shall follow a common format in most of the procedures, so a few words now will help in understanding the other procedures. The variable r% is used to hold the length (number of characters) of the string of text, while s% is the number of spaces to the left or right of the string when positioned centrally across the screen.

As the message is scrolled from the right, the procedure must manage to achieve two things. The number of characters to be displayed will increase by one each time the string moves to the left. Second, once the whole message is on the screen, but continuing to move left, the routine must ensure that the right-most character is replaced by a space at each step.

The last point is achieved by simply appending a space to the end of the string (line 120). As the string is repeatedly written one position further right each time we go round the FOR-NEXT loop, this trailing space will automatically overwrite the last visible character. The task of selecting the correct number of characters for display is handled by the MID\$ statement in the loop, which uses the loop control variable loop% as a counter.

Finally, the REPEAT-UNTIL loop at line 160 simply controls the rate of scrolling. Increase the value given to slow scrolling down, or reduce it to speed it up.

If you are using mode 7 for a title screen or similar, then the following variation of our first procedure may be used to scroll a double-height title onto the screen.

```
200 DEF PROCdisplay2(a$,y%)
210 cha$=CHR$(141)+CHR$(134)
220 r%=LEN(a$):s%=(36-r%)/2
230 a$=a$+CHR$32
240 PRINTTAB(0,y%)cha$'cha$
250 FOR loop%=0 TO r%+s%
260 ch$=MID$(a$,1,loop%)
270 PRINTTAB(38-loop%,y%)ch$
280 PRINTTAB(38-loop%,y%+1)ch$
290 TIME=0:REPEAT UNTIL TIME>8
300 NEXT
310 ENDPROC
```

Since mode 7 is always a 40-column screen this is assumed by the procedure, and just two parameters are needed, the string to be scrolled and the screen line to be used. There are two differences with our previous routine. Line 210 assigns two teletext control characters (for double height and cyan text) to the variable *cha\$*, and line 240 places these on the two lines used at the left-hand edge of the screen. All text on those two lines will henceforth appear in this format.

Second, the text itself is printed twice, once on each of the two lines (in the program lines 270 and 280). To some extent the end result is less satisfactory because of the way in which the double height text is created, and the extra work involved.

SCROLLING RIGHT

If you want to scroll the text in from the left-hand side of the screen, the task is very similar, though with some differences. The necessary code is contained in PROCdisplay3, and again is written for mode 7 double-height text.

```

400 DEF PROCdisplay3(a$,y%)
410 cha$=CHR$(141)+CHR$(134)
420 r%=LEN(a$):s%=(36-r%)/2
430 a$=STRING$(s%,CHR$32)+a$
440 PRINTTAB(0,y%)cha$'cha$
450 FOR loop%=0 TO r%+s%
460 ch1$=MID$(a$,r%+s%+1-loop%,loop%)
470 PRINTTAB(3,y%)ch1$
480 PRINTTAB(3,y%+1)ch1$
490 TIME=0:REPEAT UNTIL TIME>8
500 NEXT
510 ENDPROC

```

Because the text now scrolls from left to right, extra space is added on to the *left-hand* end of the specified string (line 430). And not just one space but sufficient to position the character string eventually in the centre of the screen. The loop extracts an increasing number of characters each time it is repeated, but the starting point for displaying the text remains constant at the left hand-side of the screen (but not overwriting the teletext control codes already placed there). The procedure could be easily simplified for single height text in this and other modes. Again, many detailed variations can be derived by experimenting with this procedure.

BOTH TOGETHER

Why not combine both ideas together? With mode 7 double-height text you can arrange for

the top half to scroll one way and the bottom half the other. This not only adds visual interest but means that the title or message is largely unreadable until the last minute. PROCdisplay4 is a suitable procedure for this.

```

600 DEFPROCdisplay4(a$,y%)
610 cha$=CHR$(141)+CHR$(134)
620 r%=LEN(a$):s%=(36-r%)/2
630 a1$=a$+CHR$32
640 a2$=STRING$(y,CHR$32)+a$
650 PRINTTAB(0,y%)cha$'cha$
660 FOR loop%=0 TO r%+s%
670 ch1$=MID$(a1$,1,loop%)
680 ch2$=MID$(a2$,r%+s%+1-loop%,loop%)
690 PRINTTAB(38-loop%,y%)ch1$
700 PRINTTAB(3,y%+1)ch2$
710 TIME=0:REPEAT UNTIL TIME>8
720 NEXT
730 ENDPROC

```

You should be able to follow this one without much difficulty. The trick, of course, is to ensure that both halves of the text end up in the same position on the screen.

ONE AT A TIME

The next routine produces a particularly effective display. Rather than scroll the whole message, it scrolls each letter in turn across the screen. Because of the shortening distance travelled by the individual characters, the letters appear to move faster and faster. A procedure for this is given below as PROCdisplay5.

```

800 DEF PROCdisplay5(a$,y%)
810 cha$=CHR$(141)+CHR$(134)
820 PRINTTAB(0,y%)cha$'cha$
830 r%=1
840 REPEAT
850 ch$=MID$(a$,r%,1)+CHR$32
860 IF ch$=" " THEN 720
870 FOR loop%=38 TO r%+5 STEP -1
880 PRINTTAB(loop%,y%)ch$
890 PRINTTAB(loop%,y%+1)ch$
900 t%=INKEY(2)
910 NEXT loop%
920 r%=r%+1
930 UNTIL r%=LEN(a$)+1
940 ENDPROC

```

Two loops are involved this time. The inner loop, using a FOR-NEXT combination, scrolls each letter in turn from right to left (note how a single space is added to the end of each character - line 850 - to overwrite the previous position as the character scrolls left). The FOR-NEXT loop is

embedded in a REPEAT-UNTIL loop which repeats the scrolling process for each letter in turn.

There are two points of particular note. Line 860 checks for a space in the character string (literally two spaces because of the extra trailing space added). If this is found the scrolling process is ignored for that letter. Without this, the procedure would scroll each space character just as for any visible character, but this results in an unnatural delay. If you are not sure try omitting line 860 and note the effect.

The second difference lies in the delay loop at line 900. Using the pseudo-variable TIME is not of much use for values under about 5 centi-seconds because of the time taken to execute the instructions themselves. INKEY provides a better delay in this case, and provides sufficient fine tuning on this occasion.

THE MOVING FINGER WRITES

The next procedure provides a quite different approach to scrolling text, again coded for mode 7 double-height text. An asterisk moves across the screen revealing the text of the message in its wake. The result is quite effective. The coding is contained in the procedure PROCdisplay6.

```
1000 DEF PROCdisplay6(a$,c%,y%)
1010 r%=LEN(a$):s%=(36-r%)/2
1020 cha$=CHR$(141)+CHR$(128+c%)
1030 PRINTTAB(0,y%)cha$'cha$
1050 FOR i%=1 TO r%
1060 PRINTTAB(s%+i%,y%) "*"
1070 PRINTTAB(s%+i%,y%+1) "*"
1080 TIME=0:REPEAT UNTIL TIME>8
1090 ch$=MID$(a$,i%,1)
1100 PRINTTAB(s%+i%,y%)ch$
1110 PRINTTAB(s%+i%,y%+1)ch$
1120 NEXT i%
1130 ENDPROC
```

Again a loop is at the heart of the procedure. On each pass, the program must display the asterisk in the next character position, and reveal (by displaying) the character previously hidden. However, all is not quite as you might expect. The loop appears to place both the asterisk and the current character in the same position (the PRINTTAB statements have the same column arguments).

The key is the delay at line 1080. The program displays an asterisk and then pauses. When it

continues, it replaces the asterisk by a character from the message, goes round the loop, and displays a new asterisk in the next character position. The consequence of this is that when the loop terminates, the last character displayed is the last character of the message, and with no further ado the asterisk has completely disappeared.


Note that this procedure incorporates an additional parameter c%. This allows a calling program to specify the colour of mode 7 text, by giving a number in the range 1 to 7. This feature could be similarly incorporated in any of the other procedures listed here in the same kind of way. Remember, though, to make allowance for any teletext control codes at the side of the screen.

DROPPING ONE'S AITCHES

The final procedure shows how it is possible to move characters around the screen in fancy ways as a message is built up. The procedure is listed below.

```
1200 DEF PROCdisplay7(a$,y%)
1210 r%=LEN(a$):x%=1:p%=4
1220 REPEAT
1230 z%=30:ch$=MID$(a$,x%,1)
1240 IF MID$(a$,x%,1)="" THEN 1360
1250 FOR i%= 5 TO 10
1260 PRINTTAB(z%+1,i%-1);" "
1270 z%=z%-1
1280 PRINTTAB(z%+1,i%-1);ch$
1290 t%=INKEY(p%)
1300 PRINTTAB(z%+1,i%-1);" "
1310 NEXT
1320 FOR i%=z% TO x% STEP -1
1330 PRINTTAB(i%,y%);ch$;" "
1340 t%=INKEY(p%)
1350 NEXT
1360 x%=x%+1
1370 UNTIL x%>r%+1
1380 ENDPROC
```

If you call this procedure, you will see the characters 'drop' down the screen before moving left to form the text. A REPEAT-UNTIL loop is executed once for each character, with one FOR-NEXT loop handling the movement down the screen, and a second scrolling the character left. Again spaces are detected and omitted from this process.

There are many more ways in which characters could be moved around, but the procedures listed here, and demonstrated on the magazine disc/tape, should give you plenty of ideas. 

Too early to think of Christmas shopping?

Not when it comes to our show!

New Horticultural Hall
Greycoat Street
London SW1



10am-6pm Fri, Nov 11
10am-6pm Sat, Nov 12
10am-4pm Sun, Nov 13

This show has it all!

LOADSA exhibitors
(around 70)
LOADSA hardware
LOADSA software
LOADSA new products
LOADSA games
LOADSA happenings
LOADSA technical
advice

... and most
important of all for
you, the visitor –
LOADSA BARGAINS!

With hundreds of
special show offers
available you could
end up a financial
winner.

You can even save
£1 a head before you
get there by using this
advanced ticket form.

The 1988 Innovation Awards

- Take a stroll down Innovation Row – a brand new show feature area, specially constructed for the event.
- See the grand finalists displaying their innovations in public for the first time.
- Help Pick The Winners. You will be able to cast a vote in both categories of the awards – the BBC Micro and the Archimedes.

No matter which Acorn machine you use – from the Electron up to the top of the range Archimedes – you'll find just what you are looking for at the show.

All the leading companies servicing each sector of the Acorn market will be out in force to demonstrate their latest developments.

Traditionally the liveliest Acorn event of the year, this pre-Christmas show is the one you can't afford to miss.

Advance ticket order	
Please supply:	
<input type="checkbox"/> Adult tickets at £4 (save £1) £	Admission at door: £5 (adults) £3.50 (under 16s)
<input type="checkbox"/> Under-16s tickets at £2.50 (save £1) £	Advance ticket orders must be received by November 2, 1988
<input type="checkbox"/> Cheque enclosed made payable to Database Exhibitions	
Total £	
Please debit my credit card account: <input type="checkbox"/> Access <input type="checkbox"/> Visa Expiry date: <input type="text"/> / <input type="text"/>	
Name Address	
Signed	
Post to: Database Exhibitions, Europa House, Adlington Park, Adlington, Macclesfield SK10 4NP.	
DATABASE EXHIBITIONS	

How to get there

Underground: The nearest tube stations are VICTORIA (Victoria, District & Circle Lines). St JAMES'S PARK (District & Circle Lines) and PIMLICO (Victoria Line).

By British Rail: VICTORIA STATION. The halls are a 10 minute walk from the station.

By Bus: 11, 24, 29, 70, 76 and Red Arrow 507 to Victoria Street – alight Army and Navy Stores.

PHONE ORDERS: Ring Show Hotline: 0625 879920
PRESTEL ORDERS: KEY *89, THEN 614568383
MICROLINK ORDERS: MAILBOX 72:MAG001
Please quote credit card number and full address

CONTEX

BANK MANAGER (for disc systems only)

The most advanced and versatile personal bank management program available for all BBC computers. Consistently acclaimed!

"data entry is a delight...professional...excellent product" - Micro User April 86.

Enter cheques and receipts. Standing orders any frequency. Automatic date sequencing. Reconcile statements. Search, amend and delete. Unreconcile. Move forwards or backwards. Analyse expenditure. Forwards cash flow forecast. Up to 36 simultaneous 'bank' (bank credit cards, savings, cash) accounts online simultaneously, inter-account transfers, 9999 standing orders, 99 analysis categories. 12 actual and 12 budgets per category, over 4,000 postings on an 80k disc. Reports to screen or printer. Graphics. Foreign currency support. Password. File recovery. Auto exec. file. Field editing. Itemised look ahead. Programmable report writer.

Standard Bank Manager for the BBC B, B+ £17.50

BANK MANAGER MASTER £22.50

Additional facilities for the Master and Master Compact.

BANK MANAGER BUSINESS UTILITIES £12.00

Adds trial balance and spreadsheet reports to Bank Manager.

ALL PRICES FULLY INCLUSIVE, FREE POSTAGE WITHIN UK

Enquiries and Access Credit card orders telephone 023 03 347

Please state disc format size

CONTEX COMPUTING

15 Woodlands Close, Cople, Bedford MK44 3UE

Cumana 40T disc drive including power supplies. Advanced Elite Filing System. Manuals included. £50. Tel. (0705) 527957 after 6pm.

BBC Master 128 in original packaging only £335. Tel. Bedford (0234) 67067 eves.

Master 65C102 6502 Turbo Upgrade £75. BEEBUG Toolkit + ROMs originals boxed with instructions. Demon Modem complete with ROM (no instr.) £30. Acorn Prestel adaptor inc. software £50. Tel.(0272) 874082 eves. and weekends.

Double 3.5" disc drive £60. Watford 32k shadow RAM card £40. Watford DDFS kit + manual £30. Basic accelerator ROM + disk £20, Wordwise plus £20. InterSheet £20. Tel. 01-730 5054.

Cumana DS DD 80T disk drive, with own PSU £75 ono. Microvitec Cub 1451 colour monitor TTL, PAL + audio inputs £140 ono. ROMs, all original with full documentation. Acornsoft Pascal, with stand alone generator £45. AMX Superart £35. Acornsoft Logo £35. Peartree Artist £20. InterWord £25. Games on 5.25" disc with original documentation: Repton 3, Codename Droid, Bone Cruncher, Crazy Rider, £20 the lot. Acornsoft Database £10. Tel. 01-737 6179.

Personal Ads

Torch Z80 second processor and software £100. 6502 second processor £65. Green screen monitor £50. Microvitec colour monitor £130. Tel (0634) 241237.

1770 Disc Interface £35, Allophone speech synthesiser (not Acorn) £15. Both unused, in mint condition. Tel. (0294) 52250, after 6 p.m.

Epson MX-80 III F/T printer, with cables and manual, £95. Tel. (09274) 24063.

Master 128 with Master ROM. Akhster Dual 40/80T switchable double 800k drive with own PSU. Microvitec High Resolution Colour 1451 monitor. Centronics GLP printer and cable, manuals. Many programs on disc. All for £550 o.n.o. Will not split. Sevenoaks area, Tel. (0732) 884940 evenings.

Viglen double-sided 48/80T switchable disc drive, powered from BBC, as new, £70. Wanted: 128k Eproms and 32k/128k RAM chips-top prices paid. Tel (0324) 558692.

SEGA Master System for sale including 2 joysticks, games worth £95, still under guarantee. Still boxed. Excellent condition. Will sell for £120 o.n.o. Phone Chelmsford (0245) 268990.

Digitiser (Watford) and service ROM. Little used and still boxed. Includes Instruction Book. £70.

BBC B View2.1, Super Art firmware, DFS, Technomatic 40/80T disk drive and Brother HR10 printer, all cables, the complete word-processing package, with manuals, £350. Tel. Skipton (0756) 69293.

Did you buy a graphics extension ROM from me in the June issue of Beebug. I have found the examples tape, but lost your address. Please phone me and I'll send it to you. Tel.(01-737) 6179.

Master 512 with DOS Plus, mouse and GEM software £420. Akhter UF twin 40/80T disc drive in plinth £150. Zenith 12" hi-res green screen monitor on tilt & swivel base £40. Master Reference Manuals 1 & 2 £15. Digital Research DOS Plus Manual £10. All leads and some software on both BBC and MS-DOS formats. Tel.(0792) 703005 day or (044128) 4461 after 6pm.

AMX Pagemaker £25. AMX Mouse package £25. ViewStore £25. All as new. Tel.(0736) 63918.

256k Solidisk Sideways RAM/ROM with 4MHz 65C02 and 256k Manager ROM (includes 200k silicon disc) plus Solidisk 2.2 DFS (unlimited number of files per disc) £80. New Interword ROM with clean registration card £30. Slave+ ROM £10. Beebugsoft Toolkit ROM £10. Tel. (0892) 21021 after 6pm.

DABS PRESS

Dabhand User News

Devoted to the Serious Expert User

David Atherton and Bruce Smith bring you the latest in Expert Software and Dabhand Guides for your BBC and Master micros.

If you have a printer then we believe HyperDriver will be one of the most significant purchases you can ever make.

Here's what Beebug said about our products in the March 1988 issue:

HyperDriver: "...an ingenious blessing.. a million other good design features..."

MOS Plus: "...an excellent product."

Master Emulation ROM: "...the whole system FEELS just like a Master...the most impressive implementation is an almost complete emulation of the temporary filing system."

Send or phone for our free 32 page catalogue.

HyperDriver: Printer Power

Possibly the most significant purchase you can make

HyperDriver isn't just another printer ROM - it's the *ultimate* one. It's absurdly easy to use and provides you with so many of the facilities missing from your current printer orientated software including: on-screen preview, CRT graphics, NLQ font, VIEW printer driver and user definable macros to name but a few.

No matter what you use your printer for, word processing, spreadsheets, databases, programming, you will have in excess of 80 * commands instantly available.

We really don't know how to cram all the facts in here, so read Geoff Bains review of HyperDriver in the March 1988 issue, pages 47/48, or ask for our latest catalogue.

Master Emulation ROM

Amazing as it may seem, if you own a BBC B or B+ then installing MER will give it virtually all the software features of a Master 128! See the Beebug review in the March 1988 issue, pages 28/29.

MOS Plus

MOS Plus is a must for all Master 128 owners not least because it fixes those irritating bugs in the OS and adds many essential extras such as *SRLOAD, *FORMAT and *VERIFY in ROM. See Beebug March 1988 pages 44/45.

Prices

Beebug members can obtain their normal 5% discount on these Dabs Press products simply by quoting their membership number (prices in brackets).

HyperDriver: ROM £29.95 (£28.52), SWR £24.95 (£23.76)

MOS Plus: ROM £12.95 (£12.33), SWR £7.95 (£7.57)

MER: ROM £19.95 (£19), SWR £14.95 (£14.24)

Orders

Send cheques, POs, official orders to the address below, or quote your Access/Visa card number and expiry date. Credit card orders accepted by phone or mailbox. P&P free in UK. Elsewhere add £2 or £10 airmail. 3.5" ADFS disc £2 extra please state if required.

Dabs Press, 76 Gardner Road,
Prestwich, Manchester, M25 7HU
Phone: 061-773-2413
Prestel: 942876210
BT Gold: 72:MAG11596

DABS PRESS

Some of the Greatest Games Ever... ...for as little as £1 each!



These compilations include some of the greatest games ever produced for the BBC Micro and Acorn Electron, including:

Planetoid One of Acornsoft's top games. "It's fast and fun, annoying and addictive. In fact, it's one of the classic micro arcade games". - Electron User
Citadel A fascinating arcade-adventure; features over 100 beautifully detailed screens of action. "The game is extremely good. Well worth the cash." ... Computer Gamer

Repton 3 Probably the best-loved of all of Superior's games. The endearing hero, Repton, features in a superb strategic game which includes character and screen designers enabling you to create your own scenarios. "This is top quality; arcade action at its very best." ... A & B Computing
 And many, many more!

SPECIAL OFFER - 3 COMPILATIONS FOR THE PRICE OF 2

For every three compilations ordered directly from Superior Software, you will be charged for only two.

For example, if you order the BBC Micro Cassette versions of **THE SUPERIOR COLLECTION 1, THE SUPERIOR COLLECTION 2 and PLAY IT AGAIN SAM** (20 top games in total), you will pay for only two of the compilations, that is, £19.90.

HOW TO ORDER

Please enter the quantities required in the boxes below and send the completed coupon, with a cheque, postal order or credit card details, to: Superior Software Ltd, Regent House, Skinner Lane, Leeds LS7 1AX.
 We will also be pleased to accept telephone orders, or mail orders, without using the coupon, but please clearly indicate that your order is for the "Three Compilations for the Price of Two Special Offer".
 Please note that if three compilations are ordered that differ in price, the free compilation will be the cheapest of those ordered.

TITLES	X - Not available in this format	BBC Micro Cassette £9.95	BBC Micro 5 1/4" Disc £11.95	Master Compact 3 1/2" Disc £14.95	Acorn Electron Cassette £9.95	Acorn Electron 5 1/4" Disc £11.95	Acorn Electron 3 1/2" Disc £14.95
THE ACORNSOFT HITS VOLUME 1 Magic Mushrooms, Planetoid, Maze, Rocket Raid (On the Electron version, Monsters is in place of Rocket Raid)						X	X
THE ACORNSOFT HITS VOLUME 2 Slingshot Command, Arcadons, Meteors, Labyrinth (On the Electron version, Snooker is in place of Labyrinth)						X	X
THE SUPERIOR COLLECTION VOLUME 1 Synchro, Repton, Karate Combat, Star Striker, Avirlit, BMX on the Moon, Wallaby, Smash and Grab					X	X	X
THE SUPERIOR COLLECTION VOLUME 2 Kix, Repton 2, Deathstar, Space Pilot, Missile Strike, Battle Tank, Crazy Painter, Overdrive					X	X	X
THE SUPERIOR COLLECTION VOLUME 3 Synchro, Repton, Repton 2, Karate Combat, Deathstar, Mr Wiz, Smash and Grab, Overdrive		X	X	X		X	
PLAY IT AGAIN SAM Citadel, Thrust, Stryker's Run, Ravenskull							
PLAY IT AGAIN SAM 2 Repton 3, Crazye Rider, Galactica, Codename: Droid						X	X

(BBC Micro games are compatible with the BBC B, B+ and Master series computers)

**SUPERIOR
SOFTWARE**
Limited

ACORNSOFT

(Acornsoft is a registered trademark of Acorn Computers Ltd. Superior Software Ltd is a registered user)
 Superior Software Ltd, Regent House, Skinner Lane, Leeds LS7 1AX. Telephone: (0532) 459453

Please make
all cheques
payable to
Superior
Software Ltd.



24 HOUR TELEPHONE
ANSWERING SERVICE FOR ORDERS



OUR GUARANTEE
 • All mail orders are despatched within 24 hours by first class post
 • Postage and packing is free
 • Faulty cassettes and discs will be replaced immediately
 (This does not affect your statutory rights)

Name _____
 Address _____

*I enclose a cheque/postal order for _____
 *Please charge to my Access/Visa card.

My card number is: _____

Signature _____

*DELETE AS APPROPRIATE

50

Security ROMs Reviewed

Recent months have seen a number of security ROMs come onto the market with the sole purpose of making the machine in which they are installed unusable by anyone except the owner. Lance Allison reviews two such devices.

Like most expensive electrical equipment, computers are very open to being stolen, be it from an educational institution or from home. Although the only way of actually preventing the computer from being removed is to physically secure it down, an alternative method is to make it useless to anyone should it be stolen. This philosophy is applied to most expensive car stereo equipment nowadays.

Product	ComputerLock
Supplier	ComputerLock 7 Ganners Grove, Bramley, Leeds LS13 2PW. Tel. (0532) 566314
Price	£25 inc.VAT (Educational discount available)
Product	Securi-Kit
Supplier	Software Services 65 South Mossley Hill Road, Allerton, Liverpool L19 9BG. Tel. (051) 427 7894
Price	Pre-programmed ROM £9.95 inc.VAT Securi-Kit disc £12.95 inc.VAT

The two products for review are Securi-Kit from Software Services and the eponymous ComputerLock. Both of these ROMs need to be placed in the highest priority ROM socket in your machine whether it be a Master or a model B. After installation the machine will always display the owner's name and address at the top of the screen upon power-up. The user is then asked for a password in order to use the machine. This will prevent the immediate use of a stolen machine. Another useful utility that both ROMs provide is

a command that temporarily disables the machine until a password is entered. If you wish to leave the room for a couple of minutes a simple star command will make sure that no one can tamper with the machine while you are away.

Securi-Kit is supplied in two forms, either as a pre-programmed ROM or as a do-it-yourself disc. If you purchase the disc you must have access to an EPROM programmer. The disc will ask you for all the relevant information such as name, address, machine serial number, and will generate a ROM image which may then be blown onto EPROM. Using this method you may protect as many machines as you like. Alternatively you can send your details to Software Services and they will make a ROM for you.

ComputerLock require your details when the ROM is ordered and supply it already in ROM form. Besides the code protection and temporary disable facilities, ComputerLock also offers quite advanced file coding commands. These commands allow single files or whole discs to be coded up in such a fashion that only the person with knowledge of the correct password will be able to decipher them again.

A major problem with both of these devices is that they are just ROMs, and thus may be removed from the machine that they were designed to protect. The only sure method of fixing the ROM in permanently is to solder or glue it in. Although it is still possible to get past these methods the ROM will serve as an effective deterrent. ComputerLock is supplied with stickers reading 'Computer Protected With COMPUTERLOCK Theft Is Pointless'. This is an extremely good idea. Any security device will only act as a deterrent if its presence is known.

All in all both products are quite professional and do their job well, but you may dislike the need to type in a password every time you switch the machine on. However, in an educational establishment or large company, where computer theft is more likely, protection ROMs may be a good investment. The decision as to which ROM to purchase, Securi-Kit or ComputerLock, will depend entirely on whether you want the file coding facilities offered by ComputerLock. If these are not important Securi-Kit is half the price and does the job just as well.

B

FONT ROM

In the first of our new BEEBUG surveys, Geoff Bains puts through their paces six ROMs that allow you to maximise the use of your printer.

These days, many dot-matrix printers are equipped with two or more Near Letter Quality (NLQ) fonts to give you more choice and more effects in your printouts, but if you are still paying off the mortgage on an older model, you are pretty much stuck with just the one print style. However, there are a number of software packages designed to alleviate this problem.

These fall into three broad categories: those which give old, non-NLQ printers an NLQ style; those which add one or more NLQ styles to an Epson compatible printer; and those which re-define the existing NLQ characters (available for a limited range of printers, namely the Taxan KP-810 and KP-910, and the Canon PW1080 and PW1156).

FONTAID AND NLQ DESIGNER

We will look at this last category first. In order to re-define the character set within a printer, a RAM chip must be fitted inside the machine. This allows the printer to take the down-loaded character definitions (about 4000 bytes of data). Such RAM chips are cheap (about £2.50) and are simple to fit - not unlike fitting a ROM to the BBC micro. Some printers are in fact supplied with the RAM already installed.

The two programs in this category are the NLQ Designer from Watford Electronics, and Fontaid from CJE Micros. Both are supplied on ROM and fulfil similar functions. Fontaid is also available in a version for the Star NL-10 but this was not tried for this review.

First of all, new fonts can be downloaded to the printer from disc with suitable star commands.

The new NLQ font is then used just as normal NLQ, with a printer Escape code being used to choose between them.

The Fontaid disc contains 10 fonts of different sizes including one of mathematical symbols. Further fonts are available on disc. The NLQ Designer disc has 29 fonts on, all of which are standard characters.

NORMAL NLQ:

abcdefghijklmnopqrstuvwxyz

SERIF:

abcdefghijklmnopqrstuvwxyz

SQUARE:

abcdefghijklmnopqrstuvwxyz

OUTLINE:

abcdefghijklmnopqrstuvwxyz

BOTH:

abcdefghijklmnopqrstuvwxyz

BROADWAY:

abcdefghijklmnopqrstuvwxyz

SHADOW

abcdefghijklmnopqrstuvwxyz

HAND

abcdefghijklmnopqrstuvwxyz

LOGO:

Canon

Fontaid and NLQ Designer

The differences appear in the font editors. Both use a 23x18 grid on which the characters are designed or altered one by one - just like most character or sprite definers used on the Beeb. The NLQ Designer displays only 32 of the whole character set at any one time but shows these on the screen in the same detail as they appear in print, unlike Fontaid which displays the whole set at a reduced resolution.

As well as pixel editing operations, both programs allow manipulation of complete characters, and even whole character sets. NLQ Designer uses the function keys for this, Fontaid a series of single key mnemonic commands. I found the NLQ Designer editor

marginally easier to use as it allows more manipulation of whole characters, but both are quite adequate.

Both packages allow the first 31 down-loaded characters, usually reserved for foreign, accented letters and symbols, to be printed packed together in two or three lines as a large but high resolution logo. Single star commands are used to print the logo, and the printhead is then returned to the start position so as not to confuse a word processor's line count.

Once defined the characters can be printed, saved to disc, downloaded, or saved as a ROM file to blow onto EPROM for installation in the printer as a permanent alternative character set. The NLQ Designer has the sensible option of printing the characters from the editor as graphics, without downloading. This saves considerable time when you're juggling a design to get it just right.

Both programs are extremely easy to use and open up a whole new world of possible fonts for owners of this range of printers.

HYPERDRIVER AND EPSON NLQ ROM

```
WE EPSON NLQ:
ABCDEFGHIJKLMN O PQRSTU VWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890

HYPERDRIVER:
ABCDEFGHIJKLMN O PQRSTU VW
XYZabcdefghijklmnopqrstu vwxyz
1234567890
```

Hyperdriver and Epson NLQ ROM

For those with printers that do not support down-loadable fonts, the other two categories of font extender programs will be the answer. The simplest of these are the programs which give non-NLQ printers a new lease of life with an add-on NLQ style. Two programs perform this task: Watford Electronics' Epson NLQ and Dabs Press' Hyperdriver. Hyperdriver also

contains a thousand and one other printer aids (see the review in Beebug Vol.6 No.9), but for this purpose we shall consider only the NLQ add-on aspect of the package.

Both programs are supplied on ROM and both must be first initialised (whereupon they take a page of RAM for workspace). They can then be almost ignored, except to turn the effect on and off. You print in the normal way, and the software intercepts all output to the printer, converting it into a sort of running graphics dump of NLQ characters.

One difference between the two packages is the way in which the effect is switched on and off. The Epson NLQ ROM uses either star commands or the ASCII codes 129 and 193. This causes problems when some control codes are sent to the printer. The NLQ mode must first be turned off, the codes sent and then the NLQ mode turned back on again. Hyperdriver uses just star commands, both to control NLQ printing, and all the other effects.

The NLQ print itself is different between these packages. The Watford program produces NLQ print in two passes, which is identical to that produced by the Taxan and Canon range already mentioned. It also manages it at a similar speed.

Hyperdriver's NLQ is produced in three passes. This means that it is much slower to print but the results are in many ways better - darker, smoother and generally more pleasing to the eye.

The Epson NLQ ROM can produce NLQ print proportionally spaced, underlined and enlarged. Hyperdriver's NLQ is always the same size and proportionally spaced (which can be a problem at times, but if you are going to be stuck with just one option, better the proportional).

The Watford ROM certainly most closely matches the effect of a real NLQ printer. However, Hyperdriver achieves better results,

and is much easier to use with only one set of commands to learn. Hyperdriver also has the advantage of a whole host of other features and, if you already have NLQ on your printer, Hyperdriver gives you a different font, rather than just repeating what's already there.

FONTWISE AND PRINTWISE

The third category of printer font software is the full-blown multi-font package. There are two of these: Fontwise from Clares and Printwise from Beebugsoft.

Again, these packages are broadly similar. Both allow any Epson compatible printer (capable of single, double and quad density graphics), or the once-popular Shinwa CP80 in the case of Printwise, to produce high resolution NLQ characters in a variety of sizes and styles. Both packages also include a font editor, but more on those in a minute.

Unlike the other types of software looked at here, these multi-font packages are not just switched into operation and then used transparently as though they were not there. These programs are used almost as printing languages. When something is to be printed in these fonts, the programs are run and the text loaded in much like a word processor. The multi-font program then prints it out to the printer in a running graphics dump in much the same way as the add-on NLQ packages.

The original text will usually be written on a word processor. Both programs can cope with text entered in Wordwise/Wordwise Plus, View and Mini Office formats. Printwise can also cope with Inter-Word text. In addition, Printwise also has its own built-in text editor for entering short pieces (up to about 425 words) without having to go to the trouble of using the word processor.

As well as just printing out the text in a suitably impressive style, these programs are also word processor back ends in their own right. They can interpret codes embedded in the text to alter the line length, justification, centring,

indents, paging and all other manner of formatting operations, and to turn on and off printer effects such as bold print, italics, proportional spacing and so forth.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz12345678

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz1234567890

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz12345678

ABCDEFGHIJKLMNOPQRSTUVWXYZa
abcdefghijklmnopqrstuvwxyz1234567

ABCDEFGHIJKLMNOPQRSTUVWXYZa
abcdefghijklmnopqrstuvwxyz123456789

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz123456

ABCDEFGHIJKLMNOPQRSTUVWXYZa
abcdefghijklmnopqrstuvwxyz123456789

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz12345678

ABCDEFGHIJKLMNOPQRSTUVWXYZabcd
abcdefghijklmnopqrstuvwxyz1234567890

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz123456789

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz1234567

ABCDEFGHIJKLMNOPQRSTUVWXYZabcd
abcdefghijklmnopqrstuvwxyz1234567890

Fontwise

In this way these packages are half way towards full 'publishing' software. With a little imagination it is easy to use this software to create professional looking brochures and booklets. Some manual paste-up work is required and all illustrations or photographs have to be handled separately, but the

'typesetting' is all provided by the font packages.

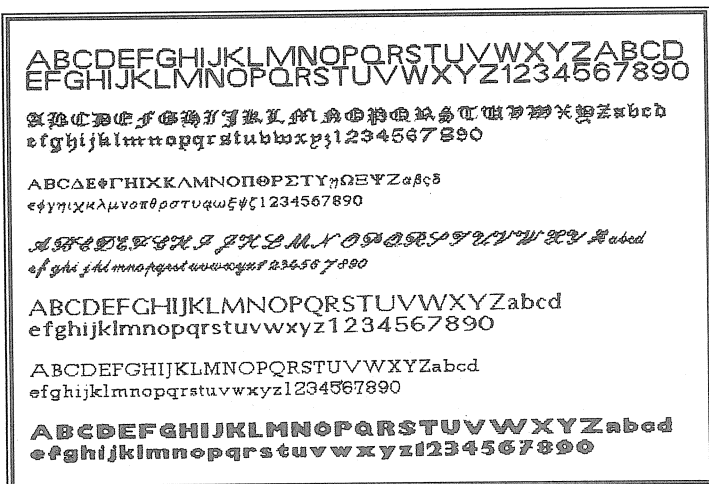
To make things easier for the majority of Beeb users, the embedded commands to specify all the formatting and printer effects are for the most part the same as those used in Wordwise Plus. Most documents prepared with Wordwise Plus can therefore be used directly in Fontwise and Printwise. A similar system applies to Inter-Word (with Printwise only) and Mini Office documents although many commands differ from the Mini Office embedded mnemonics.

For use with View, the programs differ. Printwise uses highlights 1 and 2 to surround the commands. This is simple but means the document is littered with embedded sequences which are meaningless to the wordprocessor. Fontwise gets around this problem by using View's own command structure (entered in the margin on screen) but further problems arise here from conflicting meanings between View and Fontwise commands with the same mnemonic.

The biggest difference between the packages is the range of sizes of text which they can produce. Fontwise can manage three sizes - the normal size of letters (about 8 per inch), and condensed and enlarged letters of half and twice this width.

Printwise is much more versatile. Around twenty sizes are possible with a combination of enlarged and condensed versions of different sized characters in different printer modes. However, for the full range, four separate 'fonts' are required, each covering a different range of sizes.

I must say that on the whole I prefer the Fontwise fonts. They are crisper, darker and more 'real' than the Printwise ones. However, the range of sizes available and the more versatile nature of Printwise will swing the balance for many users.



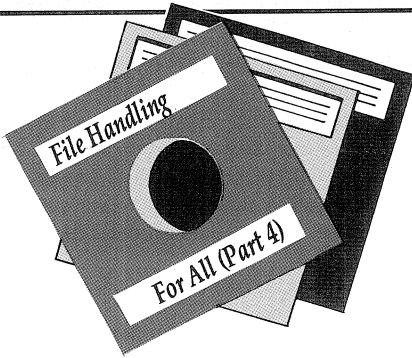
Printwise

Both Fontwise and Printwise have font editors for designing your own character styles or altering the ones provided. The editor is included in the Printwise package. The Fontwise editor used to be an extra, but it is now included.

Here again, the relative difference between the two programs is apparent. The Fontwise editor is basically just a character editor, much like any other you will see on the BBC micro. It is fast and easy to use, but each font takes up inordinate amounts of memory allowing only one to reside in memory at any one time when actually printing.

Printwise uses some clever memory saving techniques to compress the fonts into the smallest possible space. Common parts of letters are stored only once and a single letter can be stored in bits to fit between adjacent partners. This means that Printwise can hold

Continued on page 60



By Mike Williams and David Spencer

Last month we discussed in some detail the creation of a suitable file structure as part of a more professional and general approach to the writing of file handling programs. Our intention is to illustrate much of what is involved by the creation of an integrated set of procedures which you can use (and modify) as part of your own programs. So far we have provided a procedure called PROCcreatefile which may be used to create an empty data file.

We now propose to describe the procedures needed for the main features of any file handling program. These will include routines for opening and closing a file, and for adding, deleting, updating and displaying records. However, before we can proceed there is one important topic that must be covered.

DISC BUFFERING

The disc filing system which you use, whether it be the DFS or ADFS, uses a system of buffering for data transfers between disc and main memory. The buffers used (one for each file up to the allowed maximum of five) will hold 256 bytes of data using the DFS (it is handled more dynamically under the ADFS), that is the contents of one disc sector. When a program writes to a file, data from the program's own storage area is transferred to the appropriate disc buffer, and only when that buffer is full is it written out to disc.

Likewise, when a program tries to read data from disc to the program's storage area in memory, a whole sector is transferred from disc to the disc buffer, and the relevant data then supplied from the buffer to the program. Now

it is extremely unlikely that the record structure in a data file will match exactly the sector size. As a result, you may find when writing to a file, that sometimes no data transfer to disc seems to take place; the data has just not filled the buffer. The same situation may arise on input, if a previous disc read for one record has also resulted in the next record being read as well.

Although it is as well to understand what is going on, this form of disc buffering is largely transparent to the user. Most of the time it will be quite sufficient to think of data as being transferred directly between the program and disc. Of course, for maximum speed, it would be advisable to relate record size to disc buffer size, so as to maximise the amount of useful data in each buffer transfer. For most applications, the extra effort is probably not justified, and it would probably have to be paid for by more wasteful use of both disc storage and memory.

RECORD BUFFERING

However, there is another form of buffering which is almost essential to the functioning of any sensible file handling program. Most of the work any program is going to be called upon to do will relate to one complete record. We must therefore make provision for the program to be able to store a record in its own memory area. This will become the heart of our file handling system and its set-up is therefore quite important.

For example, when data is to be entered through the keyboard, it will initially be stored in this *record area*. Once a record is complete it may be transferred from the record area to the file. Likewise, a record read from a file will be placed in the record area, and the contents of the record area displayed on the screen. If we want to update a record, it will first need to be read from the file into the record area. The contents of the record area will need to be displayed on the screen, and during editing the program will need to update both the screen display and the contents of the record area in unison. Once editing of the record area is complete, its contents will be written back to disc, overwriting the original copy of that record.

Implementing such a record area is not as straightforward as it might seem at first for a program handling many different data files. There are two ways in which it might be approached: by using one or more arrays, or by *designating an area of memory and using indirection operators*. Indirection operators can be more efficient in both speed and use of memory, but it can be harder to follow the coding. For the time being at least, we will use arrays for this purpose.

```

1500 DEF PROCopen_file(name$)
1510 LOCAL I%
1520 F%=OPENUP(name$)
1530 IF F%=0 THEN PROCerror(1):ENDPROC
1540 INPUT#F%,FH%,FS%,NR%,RS%,NF%
1550 FOR I%=1 TO NF%
1560 INPUT#F%,Fname$(I%),Fwidth%(I%),Ft
type%(I%)
1570 NEXT
1580 start%=256*FH%
1590 ENDPROC

```

Now BBC Basic offers a useful facility whereby an array can be dimensioned by the value of a variable. That is to say, the size of an array can be determined by the program at run time, rather than fixed by the programmer when the program was written. Like most things there is a price to pay. An array once dimensioned cannot be re-dimensioned. This would be a limitation if we wanted to open one file for processing in some way, and then proceed to open a second file. The alternative is to use a fixed size of array, even though this will be more wasteful of storage. The size specified for the array will act as an upper limit on the number of fields possible within a record. Overall, a fixed size for this array is probably most effective, and is easier to work with for coding purposes.

There is one further point worth mentioning before we get down to writing some procedures. At the moment our plans only relate to processing one record in one file at a time. However, you may well find it useful to make provision for handling more than one record at a time by creating a two-dimensional array, thus providing a multiple buffer facility. For example, if we use last month's idea of a pointer field then we might well need to have both the initial record, and the linked record, in memory together.

Now we can get down to writing some procedures. The first one will be needed to open a file and read the File Description Record defined last time. In fact the coding will be quite similar to the PROCcreatefile procedure already written. We now just read the information previously written to the file.

We are assuming that the three arrays Fname\$(), Fwidth%() and Ftype%() have all been declared in the main program. The procedure attempts to open a file with the name supplied. If this is successful, the FDR is then read using the three arrays, and the other variables referred to last time. It is assumed that the arrays and FDR variables are treated as global variables whose values are accessible at all times throughout the program. The procedure also sets up two further global variables, F% the channel number for the file opened (which can also be used as a flag to indicate that the file is open), and start% calculated as the position of the first data record (following the FDR).

If you decide to include any other information about the file in the FDR, such as we discussed last month (date last accessed for example), then this would also be read in at this stage for use as and when required.

We have included one token error check in the procedure. This detects if the specified file can't be found (maybe you've got the wrong disc, or mis-typed the file name for example). It calls a procedure PROCerror and exits from the procedure. We are assuming that all error situations you wish to detect will be handled similarly. PROCerror, which we will not define here, would output an error message corresponding to the specified error number (here given as 1), and might well set an error flag.

This is a simple approach, and avoids the unnecessary duplication of error messages that would arise if the same error were to occur at more than one point in the program.

If you need to cater for more than one data file in use at a time, the variables F% and start% could be converted into arrays. The program just needs to keep track of a current file 'index' all the time. Of course, all our other global variables would also need an extra dimension to store the same set of information about each open file.

The next two procedures which we'll consider perform the all important task of reading or writing a single record. We will need a further array, to which we'll give the name *Record*, to implement a multi-buffer record area in memory. As all our data is assumed to be stored in string format this will have to be a string array. Our two procedures will need to designate the record number (and which memory buffer) to use:

```
2000 DEF PROCwrite_record(R%,buffer%)
2010 LOCAL I%
2020 PTR#F%=start%+RS%*R%
2030 FOR I%=1 TO NF%
2040 PRINT#F%,Record$(I%,buffer%)
2050 NEXT
2060 ENDPROC

2500 DEF PROCread_record(R%,buffer%)
2510 LOCAL I%
2520 PTR#F%=start%+RS%*R%
2530 FOR I%=1 TO NF%
2540 INPUT#F%,Record$(I%,buffer%)
2550 NEXT
2560 ENDPROC
```

In both cases the file pointer is moved to the calculated start of the specified record using PTR#. A FOR-NEXT loop deals with each field of the record in turn.

One of the first things you will want to do when writing a database program is to write a procedure to input a record and add this to the file. However, this can be quite involved as far as the screen display is concerned. It is very much up to you to decide on this for yourself. It is also closely related to the screen mode which you use. You will probably want to display the field names on the screen as a prompt for input, and you may wish to convert some of the data if you have implemented field types (dates for example). What we have coded is thus just an outline of what you might wish to do.

```
3000 DEF PROCcenter_record(buffer%)
3010 LOCAL I%:CLS
3020 FOR I%=1 TO NF%
3030 PRINT Fname$(I%)+": ";
3040 Record$(I%,buffer%)=FNinput(Ftype%
(I%),Fwidth$(I%),pad$)
3050 NEXT
3060 IF FNconfirm THEN PROCwrite_record
(NR%+1,buffer%):NR%=NR%+1
3070 ENDPROC

3100 DEF FNinput(type%,width%,p$)
. . . . .
3190 ENDPROC
```

The procedure first clears the screen (or current text window). A loop then proceeds to display the field name and input the corresponding data for each field in turn. The input of data is assumed to be handled by a further function (FNinput) with three parameters, the field type, the field width, and a specified *pad* character. The field width can be used to ensure that the data input does not exceed the maximum size for that field, while the field type can be used to automatically convert data if this feature has been implemented.

Once the data has been input and converted, it will need to be padded out so that the total number of characters is equal to the field width. The parameter pad\$ could be used to specify the pad character to be used. The resulting, justified string is then assigned to its correct position in the record area. Since a new record will be appended to the end of the file, a call to PROCwriterecord is all that is needed to add the new record to the file, and the number of records in the file (the value of NR%) is then updated accordingly. FNconfirm is a further function which would seek confirmation from the user that the new record is correct before it is added to the file.

PADDING AND STRIPPING

The choice of pad character is up to you, but whatever character you choose may not be part of any data. One solution is to use the null character (ASCII 0). This will pad out any string as required, but will be invisible if the data is displayed or printed. However, if we wish to compare the contents of a data field with any

specified string, then either the pad characters (even nulls) will need to be stripped from the data first, or the specified string will need to be padded out to the same field length. This is probably the quickest approach.

Within the record area, it is probably best to keep data in its padded form so that it is identical to the way in which it is stored in the file. Our procedures will be written on this assumption. If necessary, strip any pad characters from the data before displaying or printing it.

DISPLAYING RECORDS

Displaying a record on the screen depends much on what you are trying to achieve. Do you wish to limit the number of fields so that they may all be visible on the screen together? Perhaps you want to cater for lengthy records by splitting the screen display into several pages. Again you may be happy with a scrolling screen. If you have decided to cater for different file types you may want the program to modify the data as it is taken from the record area and displayed on the screen. We can only give a simple outline of the kind of procedure you might use.

```
3500 DEF PROCdisplay_record(R%,buffer%)
3510 LOCAL I%:CLS
3520 PROCread_record(R%,buffer%)
3520 FOR I%=1 TO NF%
3530 PRINT Fname$(I%);TAB(12);FNfield(R
ecord$(I%,buffer%),Ftype%(I%),pad$)
3540 NEXT
3550 ENDPROC

3600 DEF FNfield(data$,type%,p$)
3610 IF type%=2 THEN data$=FNinteger(data$)
3620 IF type%=3 THEN data$=FNreal(data$)
3630 IF type%=4 THEN data$=FNdate(data$)
. . . . .
3690 =FNstrip(data$,p$)
```

This follows much the same format as the procedure PROCcenter_record. The procedure calls PROCread_record to read the specified record from the file. The record is then displayed, field by field, with a field name and the corresponding data on each line. The data is

extracted from the record area, and then passed to a function called FNfield for conversion (and stripping of pad characters if required) before being displayed on the screen. We have given just an outline of what FNfield might contain. Stripping of pad characters could either be incorporated here (as shown), or in the individual conversion functions.

We will conclude this month's instalment by writing a simple function to close our data file at the end of a session. We will then have the basis of a program to create a data file, open and close a file, and to add and display records. In fact, we have put all these procedures and functions together in a simple demo program on this month's magazine disc. Here is the final function.

```
4000 DEF Fnclose_file(C)
4010 PTR#C=0
4020 PRINT#C,FH%,FS%,NR%
4040 CLOSE#C
4050 =0
```

This would be called with a line such as:

F%=Fnclose(F%)

where F% is the channel number of the file to be closed. Writing the routine as a function and calling it in this way ensures that F% is set to zero after the file has been closed, and this is consistent with the use of this variable not only as a channel number but as a flag to indicate whether or not the file is open.

The function also sets the file pointer to the start of the FDR so that the values of the first three parameters may be written back to the file. This ensures that the FDR contains (correctly) the current number of records in the file. If you have included any other information which may have been updated (such as date last accessed) then this data will also need to be written back to the file. Either write back the entire FDR, or calculate the pointer position for the particular item of data and move the file pointer there using PTR#.

We will continue our discussion of file handling in the next issue.

B

THE BANK MANAGER

If you have problems keeping up with your home accounts, you might find this package from Contex Computing very useful. Mark Joliffe explains all.

Product	The Bank Manager
Supplier	Contex Computing 15 Woodlands Close, Cople, Bedford MK44 3UE. Tel. (02303) 347
Price	£22.50 inc. VAT (Master) £17.50 inc. VAT (BBC B, B+)

The Contex Bank Manager is an accounting system aimed primarily at home users and small businesses. Versions are available for the BBC B, B+, Master 128, Compact, and Archimedes. The review version was that for the Master 128.

Bank Manager is for disc systems only, and comprises an unprotected disc and a manual. A colour monitor is an advantage, and a printer almost essential. The programs can be transferred to your own disc, and a second drive can be designated for data alone.

The features offered are extensive, and the use of the program is simple once a working disc has been created. Essentially, payments are made to or from designated accounts, each transaction being allocated a category for later analysis. Creating a working disc requires the entry of the account names and analysis groups to be used. Each account or analysis name is given a one or two digit code which is referred to in the future. In addition, each account or analysis group has a "budget" facility, which can be set up to compare present totals with previous periods, from one day to thirty-six months.

Regular payments or receipts are catered for by a 'standing order' feature, which enters these transactions automatically as they become due. Further customising adapts the program for

hardware variations or user preferences. The review version allowed multiple account sets to be accessed on the same disc, primarily for Winchester users, but useful on the large floppy disc ADPS format as well.

The programs are menu driven, with mode 7 being the usual display for data entry. Some reports appear in 80 column mode if this has been selected at set-up. All entries are in response to clear prompts, although some of the clarity will be lost without a colour monitor. Sound is used to good effect, with different notes for correct entry, deletion, and error. Option selection is by the initial letter of the keyword, or occasionally by default confirmed by pressing the Return key.

```

Amend & Reconcile                                03/07/88
Bank 1 15B Current a/c
B/f bal 582.14 Unrec 65.68 Total 647.82
All records. (X to switch)

Cheque no. 0
Amount -183.06
Description CONTRIBUTION
Posted 06/06/88
Analysis C
Reconciled NO (last record)

R reconcile U unreconcile
F forward B backward
A amend S search
I initial record L last record
D delete record Z no more
    
```

When used for the first time, a preliminary menu lists the account sets available. The initial menu is then entered, giving access to any updates on the package, utilities for sorting and recovering files, business utilities (if present), or the Bank Manager program itself. Selection of the Bank Manager option causes the configuration file and date to be read, and confirmation leads to the main menu.

This gives access to File Management, Posting of Credits and Debits, Amending and Reconciling, Reports, Graphics, or Switching of Data Sets. Each of these leads to further menus. File Management is a "Housekeeping" mode, dealing with the setting-up or altering of file parameters and configuration settings, and is seldom needed after the initial customising. The posting of credits and debits, the amending and reconciling, and the reports form the regularly-used heart of the program. The

graphics are primitive bar charts, of very doubtful worth.

Each posting is done individually on a clear mode 7 screen. After the required bank account code has been entered, the full name of that account is printed at the top of the screen together with the brought forward balance, the unreconciled total, and the account total. These three (should!) correspond to the balance at the end of the last bank statement, the total unreconciled cheques, and the last figure in your cheque-book.

Prompts for input are laid out down the screen, with default date and cheque number ready entered - the date will either be that read from the internal clock, or the date entered for the previous posting. The default cheque number will be one more than the last number entered for that account. Confirmation requires pressing Return, while overtyping clears the default and displays the new entry.

After each entry the cursor moves to the next, ending with a request for confirmation of the whole screen. Simplicity itself, with different tone beeps helping those who need to watch their typing fingers. One big irritation, which has caused many errors, is the need to prefix a credit with a minus sign. At the bottom of the Posting Screen is a function key listing: these can be pre-programmed with any often-used words.

The Amend and Reconcile screen is very similar to the Posting, without the function key list but with a further menu. Entries can be reconciled or unreconciled, altered or deleted, scanned forwards or backwards, or searched for with a variety of criteria. Scanning can be either for all, or unreconciled, entries. The prompt for this is in an unfortunate choice of white on a green background - virtually invisible. Reconciliation simply involves moving through the entries: a touch of a key gives a yellow flag to indicate that the entry agrees with your bank statement. Any error is easily corrected by selecting 'amend' and confirming each detail with Return, or by overtyping to correct. Again, very simple, very straightforward.

The Report facility gives a range of listings for all the accounts and analysis groups, with comparisons between actual and budget totals. Even deleted items can be listed. But the real gem is the Report Writer: using a short list of special commands a wide range of reports can be created to a format that suits the user. These command files are retained on disc and are called by name as required, so that more or less detailed reports can be generated when needed.

Posting		03/07/88	
B/f bal	582.14	Current a/c	65.68
		Unrec	
		Total	647.82
bank a/c 1			
Date	03/07/88	Sunday	
Cheque no.	965324		
Description	+ T.S. REEDER		
Analysis	+ F	Fire Insurance	
Amount	92.36		
OK? (Y/N/R)			
(R=write and retain data)			
Function	Keys		
0/5	1/6	2/7	3/8 4/9

A typical report could contain text as headings or explanatory comments, a listing of accounts with the current brought forward and carried forward balances, and an analysis listing with budget comparisons. Total lines can be printed with one command, and dating can be automatic. Printer commands can be entered where appropriate to produce, for instance, underlining or emphasised text.

Altogether a most smooth and pleasing package, worth the very modest cost, and backed by suppliers who have attended to queries and problems with speed, courtesy, and patience.

Product	Bank Manager Business Utility Pack 1
Supplier	Context Computing
Price	£12.00 inc. VAT

The Business Utilities provide a means of printing out Bank Manager information, either in Trial Balance or Spreadsheet format. They are supplied ready integrated with the Bank Manager suite, with the instructions on disc. The Trial Balance utility is well worth the money, but the Spreadsheet facility is of quite limited value.

B

USING ASSEMBLER PART 2

This month Lee Calcraft extends the drawing of graphics objects.

In this issue we shall be developing a generalised plotting routine, and will build on this to create line, triangle and rectangle drawing subroutines. The starting point for all this is a routine to perform the equivalent of Basic's PLOT command.

But before we begin, it is worth taking a hard look at the graphics co-ordinates which will best suit our purpose. The reason for this is that the 6502's registers are only 8 bits wide, so that unless we deal in multi-byte values, we are limited to graphics co-ordinates in the range 0 to 255. The Beeb commendably uses a 1280x1024 system for all graphics modes. But none of the modes available on the BBC micro (or indeed on the Archimedes) makes full use of this precision. The best which the Beeb can offer is 640x256 in mode 0. While all other modes provide a resolution of 320x256 or worse.

Since the 6502's registers are 8 bits wide, the greatest graphics precision that can be achieved using single registers is 256x256, and we will opt for this system here. The upshot is that our code will be much easier to write, and will perform much more quickly than it would if we had attempted to work with 16 bit numbers. Moreover the loss in resolution will be minimal. There is no loss at all in the Y direction, while the X co-ordinate is only noticeably impaired in mode 0. And even here, the fineness of any plotting will not be affected, only the precision with which a given point or line can be

specified. The only real snag is that the X and Y co-ordinates are now on a slightly different scale. The X co-ordinate is scaled down by a factor of five, while the Y co-ordinate is reduced by a factor of four. But this is very easy to live with.

Listing 1 implements a generalised plotting routine, making use of the new co-ordinate system. It is called with the plot number in the accumulator, and the X and Y co-ordinates (in the range 0 to 255) in the X and Y registers. The program also contains a routine to implement GCOL. This is called with the logic mode in the X register, and the colour number in Y. To demonstrate the two routines, the code begins with three short sequences. The first selects mode 1. The second sets colour 1 (red), and the third plots a point in that colour at the centre of the screen (co-ordinates 128,128).

Listing 1

```

10 REM General Plot
20 REM Author Lee Calcraft
30 REM Version B 0.5
40 :
50 oswrch=&FFEE
60 temp1=&70:temp2=&71
70 MODE1
80 FOR pass=0 TO 1
90 P%=&900
100 [
110 OPT pass*3
120 .start
130 \Set Mode 1
140 LDA #22:JSR oswrch
150 LDA #1:JSR oswrch
160 :
170 \Set colour 1
180 LDX #0:LDY #1:JSR gcol
190 :
200 \Plot a central point
210 LDA #69:LDX #128:LDY #128
220 JSR plot
230 RTS
240 \=====
250 .gcol
260 \Logic in X, colour in Y
270 LDA #18:JSR oswrch
280 TXA:JSR oswrch
290 TYA:JSR oswrch
300 RTS
310 \=====
320 .plot

```

```

330 \Plot code in accumulator
340 \X & Y coords in X & Y
350 \Range 0-255 (X & Y)
360 PHA
370 LDA #0:STA temp1
380 LDA #25:JSR oswrch
390 PLA:JSR oswrch
400 :
410 STX temp2:TXA
420 ASL A:ROL temp1
430 ASL A:ROL temp1
440 CLC:ADC temp2
450 BCC skip:INC temp1
460 .skip
470 JSR oswrch
480 :
490 LDA temp1:JSR oswrch
500 LDA#0:STA temp1:TYA
510 ASL A:ROL temp1
520 ASL A:ROL temp1
530 JSR oswrch
540 LDA temp1:JSR oswrch
550 RTS
560 \=====
570 ]
580 NEXT
590 :
600 CALL start

```

HOW IT WORKS

At the heart of the plotting routine is the code which multiplies the X co-ordinate by 5, and the Y co-ordinate by 4. It then uses calls to OSWRCH to send the following sequence:

VDU 25,plot no,Xlow,Xhi,Ylow,Yhi
 where Xlow and Ylow are the bottom 8 bits of the reconstituted X and Y co-ordinates (i.e. 1280x1024 style), and Xhi and Yhi are the high bytes.

Multiplying the Y co-ordinate by 4 is just a matter of shifting a register two places to the left. This is performed in lines 510 and 520. Note the use of the left shift followed by the rotate instruction (discussed in Exploring Assembler, part 9). This ensures that the bits shifted out of the top of the register are caught, and become a part of the top byte of the result. Multiplying the X co-ordinate by 5 is achieved by a similar double shift to multiply by 4, followed by adding in the original value. This is much quicker than performing a full 16-bit multiply.

LINES, TRIANGLES AND RECTANGLES

Now that we have a generalised plotting routine, we can build on this to create routines for drawing a variety of shapes. Listing 2 contains three such routines, and can draw any line, filled triangle or open rectangle. Each of the three routines makes calls to the plotting routine discussed above. As an example, take a look at the line-drawing routine at line 320 in listing 2. Each time that this is called it will draw a line between the co-ordinates x1,y1 and x2,y2. These four co-ordinates must be supplied to the routine in a four-byte parameter block. This is located at &80 onwards in zero page RAM, with x1 stored at &80, y1 at &81 and so on. The short sequence of code starting at line 150 illustrates how this works. It sets up co-ordinates 0,0 and 255,255 before executing JSR line in line 180. The result is a diagonal line from the bottom left-hand corner of the screen to the top right.

The line drawing routine itself is very simple (see line 320 onwards). It first loads the X register with the contents of &80, and the Y register with the contents of &81. It then loads the accumulator with the value 4, and jumps to the plotting subroutine. The value 4 in the accumulator causes plot code 4 to be used, so a MOVE is performed to the first of the line co-ordinates. The other two co-ordinates are picked up in the same way (line 370), and now the accumulator is loaded with the value 5, to cause the line to be drawn.

The triangle routine is also very straightforward, except that there are three sets of co-ordinates this time. These are supplied in the six bytes of zero page from &80 to &85 inclusive. Finally the rectangle routine requires just four parameters to be passed. These are the co-ordinates of the bottom left-hand corner of the rectangle and the width and height of the rectangle. They must be placed in locations &80 to &83 in the order x1, y1, width, height. Unlike the line and triangle routines, this one uses two bytes of workspace (&84 and &85).

Listing 2

```

10 REM Plot routines
20 REM line, triangle & rectangle
30 REM Author Lee Calcraft

```



```

40 REM Version B 0.5
50 :
60 oswrch=&FFEE
70 temp1=&70:temp2=&71
80 MODE7
90 FOR pass=0 TO 1
100 P%=&900
110 [
120 OPT pass*3
130 .start
140 :
150 \Draw a line
160 LDA #0:STA &80:STA &81
170 LDA #255:STA &82:STA &83
180 JSR line
190 :
200 \Draw a triangle
210 LDA #0:STA &80:STA &81
220 LDA #128:STA &82:STA &83
230 LDA #255:STA &84:LDA #0:STA &85
240 JSR triangle
250 :
260 \Draw a rectangle
270 LDA #0:STA &80:LDA #120:STA &81
280 STA &82:STA &83
290 JSR rectangle
300 RTS
310 :
320 .line
330 \x1,y1,x2,y2 at &80-&83
340 LDX &80:LDY &81
350 LDA #4
360 JSR plot
370 LDX &82:LDY &83
380 LDA #5
390 JSR plot
400 RTS
410 :
420 .triangle
430 \x1,y1,x2,y2,x3,y3 at &80-85
440 LDX &80:LDY &81
450 LDA #4
460 JSR plot
470 LDX &82:LDY &83
480 LDA #4:JSR plot
490 LDX &84:LDY &85
500 LDA #85:JSR plot
510 RTS
520 :
530 .rectangle
540 \x1,y1,width,height at &80-83
550 LDA &80:CLC
560 ADC &82:STA &84
570 LDA &81:CLC
580 ADC &83:STA &85
590 :

```

```

600 LDX &80:LDY &81
610 LDA #4:JSR plot
620 LDY &85:LDA #5:JSR plot
630 LDX &84:LDA #5:JSR plot
640 LDY &81:LDA #5:JSR plot
650 LDX &80:LDA #5:JSR plot
660 RTS
670 :
680 \=====
690 .plot
700 \Plot code in accumulator
710 \X & Y coords in X & Y
720 \Range 0-255 (X & Y)
730 PHA
740 LDA #0:STA temp1
750 LDA #25:JSR oswrch
760 PLA:JSR oswrch
770 :
780 STX temp2:TXA
790 ASL A:ROL temp1
800 ASL A:ROL temp1
810 CLC:ADC temp2
820 BCC skip:INC temp1
830 .skip
840 JSR oswrch
850 :
860 LDA temp1:JSR oswrch
870 LDA#0:STA temp1:TYA
880 ASL A:ROL temp1
890 ASL A:ROL temp1
900 JSR oswrch
910 LDA temp1:JSR oswrch
920 RTS
930 \=====
940 ]
950 NEXT
960 MODE1
970 CALL start

```

FLEXIBLE CALLING

Listing 3 provides a more taxing example of the use of the rectangle drawing routine, using it to generate a pattern made up from 120 small rectangles, as you will see when you run the program.

The whole point of using generalised plotting and drawing routines is that they may be called repeatedly, and whenever required, with parameters calculated in real time (rather than using previously determined parameters). If we are to do this, we must ensure that calling our drawing routines does not corrupt the processor's registers. This is achieved by

pushing them on to the stack at the start of the routine, and pulling them off at the end. Listing 3 contains a new version of the rectangle-drawing code which preserves registers in this way. Line 340 pushes all three registers on to the stack, and line 430 removes them again once the drawing is complete.

This permits us to use the X and Y registers as loop counters in the code which calls the rectangle subroutine. The code for generating the 120 rectangles appears from line 160 onwards. It begins by initialising locations &80 to &83 with the start co-ordinates of the first rectangle (0,0), and its size (xsize by ysize, set up in line 80 as 20x16). In lines 200 and 220 the two loop counters are loaded with their initial values (12 and 10 respectively). This will cause a 10x12 grid of boxes to be drawn.

For each cycle of the X-loop, a single rectangle is drawn, then location &80 is incremented to give a new X co-ordinate for the next plot. The X register is decremented, and the loop repeated until the X register reaches zero. At this point the next cycle of the Y loop is begun, producing a second row of boxes, and so on until the Y-counter reaches zero.

Listing 3

```

10 REM Multiple rectangles
20 REM Author Lee Calcraft
30 REM Version B 0.6
40 :
50 oswrch=&FFEE
60 temp1=&70:temp2=&71
70 xinc=24:yinc=20
80 xsize=20:ysize=16
90 MODE7
100 FOR pass=0 TO 1
110 P%=&900
120 [
130 OPT pass*3
140 .start
150 :
160 \Draw 120 rectangle pattern
170 LDA #0:STA &80:STA &81
180 LDA #xsize:STA &82
190 LDA #ysize:STA &83
200 LDY #12
210 .yloop
220 LDX #10
230 .xloop
240 JSR rectangle

```

```

250 LDA &80:CLC:ADC #xinc:STA &80
260 DEX:BNE xloop
270 LDA #0:STA &80
280 LDA &81:CLC:ADC #yinc:STA &81
290 DEY:BNE yloop
300 RTS
310 \=====
320 .rectangle
330 \x1,y1,width,height at &80-83
340 PHA:TXA:PHA:TYA:PHA
350 LDA &80:CLC:ADC &82:STA &84
360 LDA &81:CLC:ADC &83:STA &85
370 LDX &80:LDY &81
380 LDA #4:JSR plot
390 LDY &85:LDX &84:LDY &81
400 LDX &84:LDA #5:JSR plot
410 LDY &81:LDA #5:JSR plot
420 LDX &80:LDA #5:JSR plot
430 PLA:TAY:PLA:TAX:PLA
440 RTS
450 \=====
460 .plot
470 \Plot code in accumulator
480 \X & Y coords in X & Y
490 \Range 0-255 (X & Y)
500 PHA:LDA #0:STA temp1
510 LDA #25:JSR oswrch
520 PLA:JSR oswrch:STX temp2
530 TXA:ASL A:ROL temp1
540 ASL A:ROL temp1:CLC
550 ADC temp2:BCC skip:INC temp1
560 .skip
570 JSR oswrch:LDA temp1
580 JSR oswrch:LDA#0:STA temp1
590 TYA:ASL A:ROL temp1
600 ASL A:ROL temp1:JSR oswrch
610 LDA temp1:JSR oswrch
620 RTS
630 \=====
640 ]
650 NEXT
660 MODE1
670 CALL start

```

You will find that you can easily experiment with this routine to create different effects. You could for example alter the size of the rectangle at certain stages of the loop, or animate the rectangle by plotting and over-plotting using GCOL 3,n. There is also plenty of scope for writing routines to generate other graphics objects.

Next month we will take a look at the new topic of vectors, but we will return to graphics at a later date.

B

David Spencer continues his look at linked lists, and their use in Basic.

Last month we looked at the use of pointers, and how these could be used to implement linked lists within Basic. However, we covered only the creation of a new list, the addition of a record to the end of a list, and the reading of a list. Two further important operations which we need to consider are the addition and removal of a record in the middle of a linked list.

We will consider insertion first, for which our starting point will be a linked list in memory, and a new record to insert into the list. We will assume that this new

record A into record B, so that both now point to record C. Assuming that our pointers are four byte values at the start of each record, this could be done using:

```
!newptr=!ptr
and Figure 2 depicts this.
```

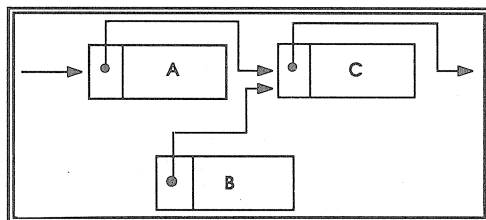


Figure 2

Finally, the link from record A has to be redirected to point to record B, which could be done using:

```
!ptr=newptr
```

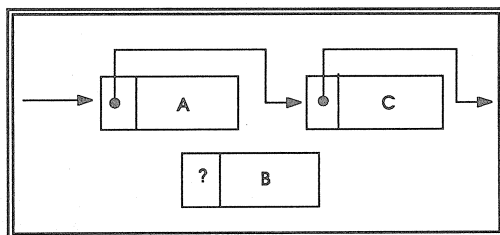


Figure 1

record already has a pointer field, but its contents are undefined. We will also assume that there is a pointer in the program called 'ptr' which points to the record (referred to as A) preceding the point of insertion, and another pointer called 'newptr' which points to the new record (referred to as B). Record C is the record that is currently after A. This is shown in figure 1.

The first thing to do is to copy the pointer field of

This is shown in figure 3, and you can see that the new record is now linked into the list. It is important to realise that the records will not normally be in order in memory. The order of the list is determined solely by the pointers, regardless of where the

actual records are.

Deletion is even easier. Consider deleting record B from figure 3. This is easily

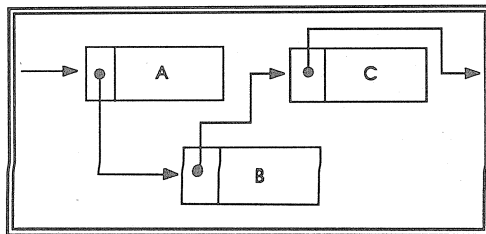


Figure 3

accomplished by taking the link pointer from record B, which points to record C, and storing this value in the link pointer of record A. This

makes record A point to record C, effectively by-passing record B, although it is still in memory. In Basic this could be done using:

```
!ptr=(!ptr)
```

This returns to the state of figure 2.

Incidentally, these techniques can also be used to insert and remove records from the *end* of a linked list. This is because at no point is the record after the one being deleted or inserted accessed. All that is used is a pointer to that record, and if you are at the end of the list, this will simply have a null value.

LIST HEADS

One of the problems that occurs with linked lists is the handling of an empty list, in other words, one containing no records. If you adopt the method we showed last month, with a separate pointer that points to the start of the list, then an empty list will be indicated by this pointer having a null value. While this method may seem straightforward, it can cause a lot of unnecessary work when accessing the list. For example, consider a linked list of records that will be stored in memory somewhere, with the first record pointer to be a pointer called 'FirstPtr'. Now assume that we have a record in memory at location 'MemLoc' to add to the end of the list. A fragment of code to do this might look like:

```
IF FirstPtr=0 THEN FirstPtr=MemLoc
ELSE PROCinsert(FirstPtr)
:
:
DEF PROCinsert(Ptr)
IF !Ptr=0 THEN !Ptr=MemLoc ELSE
  PROCinsert(!Ptr)
ENDPROC
```

The action of this code is to go through the list until a null link pointer is found, and then to set that pointer to point to the new record. We have used a recursive technique to work our way through the list (see the Workshop in Vol.7 No.2 for more information on recursion). This simply treats the list as a series of smaller lists. Each time PROCinsert is called, the list is effectively one record shorter.

You can see that we need to pay special attention to the case of an empty list. In such circumstances we cannot just set the link pointer of the last record in the list to point to the new record, because there is no last record.

Instead, we have to alter the value of 'FirstPtr' which is a pointer to the whole list.

There is, however, an easy way round this problem. This is to place a dummy record at the start of the list, and make the link pointer in this dummy record point to the first real record. Such a dummy record is called a 'List Head', because it appears at the head of the list. A list head doesn't have to be the same type of record as the rest of the list, because all that is needed is the pointer field. The list head is created at the start of execution, before any manipulation of its list takes places, and it should be set up with a null pointer. This means that the list consists of just the list head. The advantage of this, is that there is always at least one record, and so we no longer have the special case of a totally empty list to deal with.

Using this method, our operation for inserting a record at the end of the list could be coded as follows, (assuming that 'FirstPtr' now points to the list head):

```
PROCinsert(FirstPtr)
:
:
DEF PROCinsert(Ptr)
IF !Ptr=0 THEN !Ptr=MemLoc ELSE
  PROCinsert(!Ptr)
ENDPROC
```

This time, we do not need the extra code to check for an empty list, as this is handled automatically.

Listing 1 is an example program that reads words that are typed in, and stores them in a linked list in alphabetical order. If you then press Return on its own, the program switches to delete mode. Any words now entered will be deleted from the list. A further press of Return on its own will result in all the words in the list being printed out in alphabetical order. Each word is also stored with a count of the number of times that word has been entered. However, when you delete a word from the list the word is simply removed, without any reference to the number of times it has occurred.

The record format used to store each word is:

- 4 byte link pointer (Memory address)
- 4 byte integer count
- ASCII characters of word, terminated with a carriage return.

A list head is set up at the start of our memory block. This consists of just a single four byte pointer, which is initially set to zero to indicate an empty list. The procedure to insert a word starts at the list head and searches through the existing list. If the new word is already present, then its 'frequency count' is incremented. Otherwise, the search continues until a word is found that comes after the new word (alphabetically). The correct position for the new word is immediately before this record. At this stage, a record is created for the new word, and it is linked into the list. A special check has to be made for the end of the list. If we reach the end before finding a place for the new word, then the new word must be the last word alphabetically, and it is inserted at the end of the list.

The deletion routine is similar. This checks through the list until the desired word is found, and when it is, unlinks the corresponding record from the list. If the word is not found before the end of the list, then a warning message is printed, and the list is left unchanged.

One possible use for a data structure along these lines is the implementation of a simple spell checker. Rather than typing in each word, you could have a procedure that read a text file, and returned it a word at a time. By the end of the text, you would end up with a sorted list of all the words read in, along with a count of how many times each word occurred. You could then go through the list one word at a time, check each entry against the spell checker's dictionary, and if it is found remove the word from the list. After all the list has been checked, all that is left is a list of words not in the spell checkers dictionary, along with a frequency count of their usage.

MEMORY ALLOCATION

Our discussion of deleting records from a list gives rise to an interesting problem. What do you do with the deleted record? As it stands, any records we delete simply get left tucked in a corner of memory. While this is no problem with our simple examples, with a larger list where records are constantly added and deleted, we could quickly run out of memory as it is filled up with old records.

What is needed is a set of routines that can allocate chunks of memory as needed, and then clean up once records have been deleted so that the memory space can be re-used. Such a system will be developed in the next Workshop, and surprise, surprise, it will use linked lists in its operation. Also, next time, we will show how Basic uses linked lists to store its variables.

```

10 REM Program Linked List Word Sort
20 REM Version B 1.0
30 REM Author David Spencer
40 REM BEEBUG August/September 1988
50 REM Program subject to copyright
60 :
100 DIM list 1000:free=list+4
110 !list=0: REM List Head
120 PRINT "ENTER WORDS"
130 REPEAT INPUT word$
140 IF word$<>" " PROCenterword(list,wo
rd$)
150 UNTIL word$=""
160 PRINT "DELETE WORDS"
170 REPEAT INPUT word$
180 IF word$<>" " THEN IF NOT FNremovew
ord(list,word$) PRINT "Word not in list"
:VDU 7
190 UNTIL word$=""
200 ptr=list
210 REPEAT
220 ptr=!ptr
230 IF ptr THEN PRINT$(ptr+8);TAB(20);
ptr!4
240 UNTIL ptr=0
250 END
260 :
1000 DEF PROCenterword(ptr,A$)
1010 IF !ptr=0 THEN 1040
1020 IF $(!ptr+8)=A$ THEN !(!ptr+4)=!(
ptr+4)+1:ENDPROC
1030 IF $(!ptr+8)<A$ THEN PROCenterword
(!ptr,word$):ENDPROC
1040 !free=!ptr:free!4=1
1050 $(free+8)=A$
1060 !ptr=free
1070 free=free+LEN(A$)+9
1080 ENDPROC
1090 :
1100 DEF FNremoveword(ptr,A$)
1110 IF !ptr=0 THEN =FALSE
1120 IF $(!ptr+8)=A$ THEN !ptr=!(!ptr):
=TRUE
1130 =FNremoveword(!ptr,A$)

```

B

DUAL SCREEN PROGRAM DISPLAYS

Paul Pibworth presents a useful utility that will allow two programs to be examined on the screen at the same time, using a split screen display.

How many times have you found yourself in the position where you cannot exactly remember the subtle differences between two versions of the same program? This utility will allow you to examine both programs on the screen at the same time.

Simply enter the program as listed and save it as usual. When the program is run it will ask for the file names for both programs which should previously have been saved to disc or tape. Enter the appropriate names and you will be presented with two boxes, one for each program. The cursor will be placed in the top left-hand corner of the top box. Use the cursor left/right keys to increment/decrement through the line numbers.

Pressing Shift whilst using the cursor keys will prevent the Basic lines from being displayed, allowing the program to be searched through at a quicker pace. Pressing Ctrl and cursor left will take you directly to the beginning of the program. Pressing P will print the current line on the printer. Lines from the first program will appear on the left hand side of the paper while lines from the second program will appear on the right. This makes cross referencing the listings easier. You may swap between boxes by using the cursor up/down keys. The current box is signified by the presence of a >>>> next to the file name. Once you have mastered the key functions, you will find that programs may be interrogated quickly and easily. When you have finished press Escape to exit.

For those who are interested in how the program works it simply loads the two files above itself in memory. This is far faster than

any alternative method, such as using random access files. Once loaded the small machine code routine in PROCassem is used to decode Basic lines as they are needed. Because the two programs to be compared are loaded above the utility, memory is at a premium. This may limit the lengths of programs to be compared. If you find that you do not have enough memory we would suggest that you remove PROCbox and lines 230-240. If you also remove all REM statements and spaces you will be able to change line 140 to:

```
140 B%=PAGE&D00
```

If you have a utility ROM such as BEEBUGS Toolkit Plus you may compact the program (*CRUNCH on Toolkit Plus).

As a final point regarding use of memory, be careful not to insert any additional spaces when entering the listing. If you keep a copy of the utility nearby you should find it very useful indeed.

```
10 REM Program   Compare Utility
20 REM Version   B1.05
30 REM Author    Paul Pibworth
40 REM BEEBUG    August 1988
50 REM Program   subject to copyright
60 :
100 ON ERROR MODE 7:PROCError:END
110 MODE 7
120 PROCassem
130 A$=STRING$(253,CHR$32):A$="PRESS A
RROW!"
140 B%=PAGE&F00
150 INPUT""ENTER NAME OF FIRST PROG "
'N1$
160 INPUT""ENTER NAME OF SECOND PROG
"N2$
170 OSCLI"LOAD "+N1$+" "+STR$~B%
180 Z%=OPENUP N1$:C%=EXT#Z$:C%=C%DIV25
6:C%=C%*256+256+B%
190 CLOSE #Z%
200 OSCLI"LOAD "+N2$+" "+STR$~C%
210 :
220 CLS
230 PROCbox(0,2,8,39)
240 PROCbox(0,15,8,39)
250 PRINTTAB(6,0)"PROGRAM :-"N1$
260 PRINTTAB(6,13)"PROGRAM :-"N2$
270 D%=B%:Y%=B%
280 E%=C%:Z%=C%
290 *FX4,1
300 G%=FNvdu(8,35,3,1)
310 REPEAT
```

```
PROGRAM --PROG1
```

```
160 PRINT "Press space bar to re-
cord temperature." TAB(7) "or Esca
pe to calculate."
```

```
>>> PROGRAM --PROG2
```

```
160 REM insert *SP00L filename h
ere for MODE=00 dump
```

```
320 F%=GET
330 IFF%=139 VDU26:PRINTTAB(0,0) ">>>
";TAB(0,13) SPC(5):G%=FNvdu(8,35,3,1)
340 IFF%=138 VDU26:PRINTTAB(0,13) ">>>
";TAB(0,0) SPC(5):G%=FNvdu(21,35,16,2)
350 IFF%=136 AND G%=1 AND NOT INKEY-2
D%=FNback(Y%,B%):F%=137
360 IFF%=136 AND G%=1 AND INKEY-2 D%=B
%:Y%=B%
370 IFF%=136 AND G%=2 AND NOT INKEY-2
E%=FNback(Z%,C%):F%=137
380 IFF%=136 AND G%=2 AND INKEY-2 E%=C
%:Z%=C%
390 IFF%=137 AND G%=1 Y%=D%:D%=FNforwa
rd+D%
400 IFF%=137 AND G%=2 Z%=E%:E%=FNforwa
rd+E%
410 IF NOT INKEY-1 PROCdisplay
420 IFF%=80 PROCprint
430 UNTIL F%=69
440 VDU26,12:*FX4,0
450 END
460 :
1000 DEF FNvdu(H%,I%,J%,K%)
1010 VDU28,4,H%,I%,J%,31,0,0
1020 =K%
1030 :
1040 DEF FNforward
1050 IFG%=1 mem%=D% ELSE mem%=E%
1060 IFmem%?1>127 =0
1070 L%=mem%?1:M%=mem%?2:N%=mem%?3
1080 O%=L%*256+M%
1090 CLS:IFL%<&FF PRINT;O%
1100 =N%
1110 :
1120 DEF FNback(mem%,base%)
1130 IFmem%=base% =base%
1140 LOCALcount%:count%=0
1150 REPEAT:count%=count%-1
1160 UNTIL?(mem%+count%)=13 AND ?(mem%+
```

```
count%+3)=ABScount%
1170 =mem%+count%
1180 :
1190 DEF PROCdisplay
1200 IFG%=1 A$=$ (Y%+4):O%=? (Y%+1)*256+?
(Y%+2):L%=Y%?1:ELSE A$=$ (Z%+4):O%=? (Z%+1)
)*256+?(Z%+2):L%=Z%?1
1210 token%=-1
1220 IFL%=255 CLS:PRINT TAB(10)"END":EN
DPROC
1230 PRINTTAB(0,0):O%=" ";
1240 FORQ%=1TOLENA$
1250 A%=ASC MID$(A$,Q%,1)
1260 IFA%=34 token%=token%*-1
1270 IFA%>31 AND A%<127 PRINTCHR$(A%);
1280 IFA%=&8D:PRINT;(256*((ASC MID$(A$,
Q%+1,1)*16)AND192) EOR ASC MID$(A$,Q%+3,1
)))+(ASC MID$(A$,Q%+1,1)*4)AND192) EOR
ASC MID$(A$,Q%+2,1));:Q%=Q%+3:A%=0
1290 IFA%>127 AND token%=-1 CALL&900
1300 IFA%>127 AND token%<>-1 PRINTCHR$(A
%);
1310 IFA%=&F4 token%=0
1320 NEXT
1330 PRINT
1340 ENDPROC
1350 :
1360 DEF PROCprint
1370 IFG%=1 VDU2,1,27,1,108,1,5,1,27,1,
81,1,40,3
1380 IFG%=2 VDU2,1,27,1,64,1,27,1,108,1
,45,1,27,1,81,1,80,3
1390 VDU2:PROCdisplay:VDU3
1400 ENDPROC
1410 :
1420 DEF PROCassem
1430 FORpass=0 TO 2 STEP 2
1440 P%=&900
1450 {OPT pass
1460 STA&80
1470 LDA&8015
1480 CMP#50:BEQ bas2
1490 CMP#52:BEQ bas4
1500 BRK:EQU$" BASIC ROM NOT RECOGNISED
":BRK:RTS
1510 .bas2
1520 LDA#(&71-10):STA&85
1530 LDA#&80:STA&86
1540 LDA#&8A:STA&87
1550 JMPtoken
1560 .bas4
1570 LDA#(&56-10):STA&85
1580 LDA#&84:STA&86
1590 LDA#&81:STA&87
1600 .token
1610 LDA&85:STA&81:LDA&86:STA&82
```

```

1620 LDY#10
1630 .loop2
1640 LDA(&81),Y:CMF&80:BEQ found
1650 CLC:LDA&81:ADC#1:STA&81
1660 LDA&82:ADC#0:STA&82:CLC
1670 JMP loop2
1680 .found
1690 DEY
1700 LDA(&81),Y
1710 CLC:CMF&80:BCC found
1720 CLC:CMF&87:BEQ print2
1730 INY
1740 .print2
1750 INY:LDA(&81),Y
1760 CLC:CMF&7F:BCS rts
1770 JSR&FFEE:JMP print2
1780 .rts
1790 RTS
1800 ]
1810 NEXT

```

```

1820 ENDPROC
1830 :
1840 DEF PROCerror
1850 REPORT:PRINT " at line ";ERL
1860 *FX 4,0
1870 ENDPROC
1880 :
1890 DEF PROCbox(x,y,h,w)
1900 LOCAL i,j
1910 PRINTTAB(x,y);CHR$(147);CHR$(183);ST
RING$(w-2,CHR$(163));CHR$(107)
1920 FOR j=0 TO h-3
1930 PRINTTAB(x,j+y+1);CHR$(147);CHR$(181
);CHR$(135);TAB(x+w-1,j+y+1);CHR$(147);CHR$(
106)
1940 NEXT j
1950 PRINTTAB(x,y+h-1);CHR$(147);CHR$(117
);STRING$(w-2,CHR$(112));CHR$(122)
1960 ENDPROC

```

B

CROSSWORD EDITOR (Continued from page 10)

```

2870 DEF PROCcont
2880 VDU7:PRINTTAB(1,VPOS)"Press any ke
y to proceed":*FX15
2890 B=GET:ENDPROC
2900 :
2910 DEF PROCOptions
2920 LOCALX%:VDU28,1,7,12,1
2930 COLOUR131:CLS:COLOUR2
2940 FORX%= 0TO5:PRINTTAB(1)O$(X%):NEXT
2950 ENDPROC
2960 :
2970 DEF PROCspell
2980 LOCAL X%,C,ex
2990 PROCOptions:ex=FALSE
3000 REPEAT:S$=""X%=0:COLOUR2
3010 REPEAT:*FX21,0
3020 COLOUR 128:PRINTTAB(1,X%)O$(X%)
3030 C=GET:COLOUR131:PRINTTAB(1,X%)O$(X
%)
3040 IF C=139 ANDX%>0 X%=X%-1
3050 IF C=138 ANDX%<5 X%=X%+1
3060 UNTIL C=13:IF X%=5 ex=TRUE
3070 IF NOTex S$=FNgetstr
3080 IF NOTex ANDS$<>" " PROCexec(X%)
3090 PROCOptions
3100 UNTILex
3110 COLOUR129:CLS:VDU26:COLOUR130
3120 ENDPROC
3130 :
3140 DEF FNgetstr
3150 COLOUR129:CLS:VDU28,5,5,33,1
3160 COLOUR131:CLS:COLOUR2:PROCon
3170 PRINT"Search string":INPUT":S$
3180 PROCoff:COLOUR129:CLS

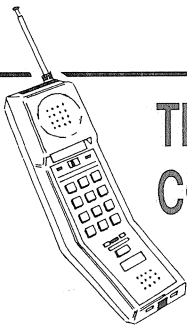
```

```

3190 =S$
3200 :
3210 DEF PROCexec(O%)
3220 VDU28,25,30,38,1,14:COLOUR131:CLS
3230 IF O%=0 OSCLI("ANAGRAM "+S$)
3240 IF O%=1 OSCLI("BROWSE "+S$):GOTO33
00
3250 IF O%=2 OSCLI("CHECK "+S$)
3260 IF O%=3 OSCLI("CROSSWORD "+S$)
3270 IF O%=4 OSCLI("FUZZY "+S$)
3280 VDU7:PRINT"Press a key":*FX15
3290 B=GET
3300 VDU15:COLOUR129:CLS:ENDPROC
3310 :
3320 DEF PROCmc
3330 FOR Z%=0TO2STEP2:P%=&A00:[OPTZ%
3340 LDA#17:JSR&FFEE:LDA#130:JSR&FFEE
3350 LDA#&FF:STA&71:LDX#0
3360 .x INX:LDY#0:.y INY:INC&71
3370 TYA:PHA:LDY&71:LDAW%,Y
3380 STA&7F:BEQzero:CMP#1:BEQone
3390 LDA#0:STA&75:JMPpl
3400 .zero LDA#0:STA&75:LDA#32:STA&7F
3410 JMPpl
3420 .one LDA#3:STA&75:LDA#255:STA&7F
3430 .pl PLA:TAY
3440 LDA#17:JSR&FFEE:LDA&75:JSR&FFEE
3450 LDA#31:JSR&FFEE
3460 TXA:CLC:ADC&454:JSR&FFEE
3470 TYA:CLC:ADC&458:JSR&FFEE
3480 LDA&7F:JSR&FFEE
3490 CPY&70:BMIY:CPX&70:BMIx
3500 RTS:]NEXT
3510 ENDPROC

```

B



THE COMMS SPOT

In this month's Comms Spot, Peter Rochford investigates what one should look for when purchasing communication equipment.

In last month's Comms Spot, I presented a list of communication terms with explanations for those new to the subject. This month I am going to explain what features to look for in terms of both the hardware and software needed for a comms set-up.

MODEMS

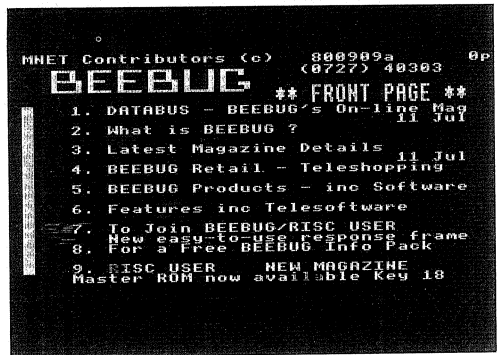
I explained what a modem is and what it does in last month's column. It is a hardware device and quite a complex piece of electronics. However, it does not need to cost the earth and prices have actually fallen recently, largely due to production of custom chips like the AMD 7910. These chips contain most of the modem's circuitry all in one package, instead of in the form of discrete components, thus saving on cost.

When choosing a modem you must first decide at what speeds you wish to communicate. 300/300 baud (V21) is standard on most modems but is rarely used now. 1200/75 (V23) is again standard on most modems and is necessary to communicate with Prestel. Telecom Gold also works at 1200/75 but can be used at 300/300. Unfortunately the latter is painfully slow and costly in terms of both your telephone bill and online time charges.

The next speed up the scale is 1200/1200 (V22). Both Prestel and Telecom Gold can operate on this standard. In the case of Prestel, V22 is not much of an advantage unless you are sending large numbers of mailboxes which you have

created off-line and then wish to upload quickly. Prestel responds, in the main, to single key presses or short keywords to request information. The difference between sending them at 1200 baud instead of 75 baud will hardly be noticeable or advantageous to the user. Neither will typing in mailbox messages while on-line be much faster at 1200 baud compared to 75 baud, as most people cannot type fast enough to benefit from the extra speed of data transmission.

When it comes to Telecom Gold, with its far more sophisticated Email system, then V22 can be much more desirable when you regularly create large text files off-line and then upload them quickly onto the system and log off, thus saving time and money.



The next most common speed offered on modems, and in use by BT on their online systems, is 2400/2400 baud (V22bis). This is twice the speed of V22 and can mean significant savings to you in terms of time and money, not to mention the sheer luxury of getting the information on your screen so fast. The penalty at the moment however, is that V22bis modems are still quite expensive at around £500. Until the price of the technology falls, as it surely will, it is beyond the reach of most domestic users.

So, having decided which speeds your modem should have, what other features should you look for? Auto-answer is one that is offered as standard on many modems. However, this will

be a little-used facility by most unless you want to set up your own bulletin board.

What about auto-dial? Definitely worth having I believe as it saves the bother of dialling manually and enables you to keep a database of your most regularly used numbers, for selection and dialling via software by means of a single keypress.

A large number of modems are now available that are Hayes compatible. I have talked before about Hayes and its advantages. Briefly, it is a standard developed in the USA whereby the modem will respond to a set of commands that enable speed selection, auto-dial and auto answer amongst other things, all under software control. The beauty of it is that the modem will work with virtually any software that allows the codes to be sent to the modem. This is a great step forward in comms, and I would not consider buying a modem that was not Hayes compatible.



SOFTWARE

This is a difficult subject to tackle and causes most people the biggest headaches. There are a large number of comms software packages around for the Beeb. The majority of them are very good, but some are downright awful.

Choosing a package can sometimes be dictated by the modem you have chosen, as the modem is designed to work with particular software to select its functions. Such was the case of the ill-

fated Demon modem and the accompanying Zromm software.

What else should you look for in a software package? It certainly should provide the facility to access both scrolling text and viewdata services. It should also allow you to communicate at all the speeds that your modem is capable of. If your modem is not Hayes compatible then it should allow you to auto-dial from your modem if that is possible. Certain packages like Commsoft have modem drivers available to allow this. It consists of an extra piece of software loaded into RAM once the main software has been called up.

In Viewdata mode the software should allow telesoftware downloading, the printing and saving of screens and off-line mailbox editing. These are, I believe, the very basics.

In scrolling text mode there should be the facility for uploading and downloading of text files, saving of output/input to disc or memory buffer, printing of all screen output, choice of screen modes and the ability to download/upload data and text using one or more of the popular error checking protocols such as Xmodem or Kermit. These again are the basics that should be included.

There is one final and particularly important criteria to mention in choosing comms software, and that is ease-of-use. You must be happy with the software and find it simple to use. If not, your on-line time will become a chore and will ultimately cost you more money as time is wasted whilst you fight to get the software to do what you want.

As with most products you purchase, try to see the modem and software working together before you buy. Not always easy to get that sort of demonstration, but well worth trying for.

Note: BEEBUG's internal modem for the Master 128 is now available for £113.05 to members, and is supplied complete with our Command ROM and an 84 page manual.

B

FONT ROM (continued from page 41)

two fonts in memory at once and these can be instantly switched between when printing (with yet another embedded command). This also means that unlike Fontwise, Printwise can print a single line in more than one font - a very useful feature.

However, this complication of the font data handling also makes the Printwise font editor less friendly than Fontwise. As ever it's a choice of ease of use against versatility.

CONCLUSION

Both Fontwise and Printwise are excellent packages in their own right and the choice between them is largely a matter of personal preferences and requirements. The same goes for the rest of these font packages too.

If it's just a straightforward NLQ you require then Watfords Epson NLQ and Hyperdriver provide simple solutions. Hyperdriver also has the advantage of offering a host of other goodies too.

For owners of the Taxan and Canon printers covered by the NLQ Designer and Fontaid ROMs then either of these provide high quality

and easily-used alternative printer styles. These make use of the printer's existing (if rarely used) features which both knit in perfectly with the normal printer effects and provide a high quality output.

However, these two ROMs cannot compete with the two multi-font programs (Fontwise and Printwise) when it comes to versatility. For truly professional output, these programs are only beaten by 'publishing' software such as Fleet Street Editor and Stop Press - both of which are considerably more complicated and expensive.

SUPPLIERS

Watford Electronics, 250 Lower High St., Watford WD1 2AN, phone (0923) 37774.

Dabs Press, 76 Gardner Road, Prestwich, Manchester M25 7HU, phone 061-773 2413.

Clares Micro Supplies, 98 Middlewich Road, Northwich, Cheshire CW9 7DA, phone (0606) 48511.

Beebugsoft, Dolphin Place, Holywell Hill, St. Albans, Herts. AL1 1EX, phone (0727) 40303.

CJE Micros, 78 Brighton Road, Worthing, West Sussex BN11 2EN, phone (0903) 213361.

B

PROGRAM	PRICE	FORMAT	PRINTERS	No. FONTS	FONT EDITOR	ENLARGED/ CONDENSED	OTHER SIZES	>1 FONT PER LINE
Epson NLQ (Watford)	£25	ROM	Epson compat.	1	no	yes	no	yes
Fontaid (CJE Micros)	£30	ROM	Taxan/ Canon/ Star	10	yes	yes	no	yes
Fontwise (Clares)	£30	disc	Epson compat.	22	yes	yes	no	no
Hyperdriver (Dabs Press)	£29.95	ROM	Epson compat.	1	no	no	no	no
NLQ Designer (Watford)	£25	ROM	Taxan/ Canon	29	yes	yes	no	yes
Printwise (Beebugsoft)	£22.50	disc	Epson/ Shinwa compat.	9	yes	yes	yes	yes

HINTS HINTS HINTS HINTS HINTS

and tips and tips and tips and tips and tips

David Spencer rounds up this month's Hints and Tips from BEEBUG members.

*** STAR HINT ***

MASTER SERIAL PORT

Andre Peters

There is a very subtle difference between the serial port on the model B, and that on the Master, which stems from the use of a different serial input chip on the two machines. When a working RS232 or RS423 device is connected to the serial port, both machines behave identically. However, the effect when the serial port is unconnected is different between the model B and the Master. On a model B, both the data input, and the Clear To Send (CTS) input, float to a logic high level. This means that no data is received by the computer on the serial port. On a Master, however, both of these lines float to a logic low. This means that the computer will continually receive Null characters on the serial port. Further, all these characters will cause what is known as a framing error, because the line is held low all the time. Normally, the operating system can cope with this, and just ignores the erroneous characters. But, some third party software written for the model B might not be able to handle this properly, and this could prevent the software from working.

Acorn say that the solution to this problem is a small hardware modification which

involves adding two resistors to the Master. Both should be 15K in value, with one connected between IC50 pin 8 and IC51 pin 6, and the other between IC50 pin8 and IC51 pin8. As this involves soldering directly to integrated circuits on the PCB, it is better to have the modification carried out by an approved dealer.

TELLING BASIC APART

Gary Blackwell

It is often very useful when writing machine code programs to be able to access routines within the Basic ROM. This can save writing large chunks of code which Basic already provides. However, one of the main problems of this is that there are several versions of Basic around, and the same routine is unlikely to be at the same address in any two versions. One way around this is to keep a table of the addresses of the routines for each version of Basic, and then pick the correct one when the program is assembled. However, you still need to find which version of Basic you have. The easiest way of doing this is to look at the copyright date in the Basic ROM. The byte at location &8015 contains the last digit of the year in ASCII. You can then write a line such as:

```
IF ?&8015=ASC"4" THEN
PROCbasicIV
```

The values to use are "1" for Basic I "2" for Basic II, "4" for Basic IV, and "6" for Basic VI (Compact).

COMPACT VIEW

John Wallace

Many users of View on the Compact do not realise that it contains a built in printer driver for Epson compatible printers. This driver is automatically selected when View is started, but if a different driver is used, the default Epson one can be re-installed by typing:

PRINTER EPSON

from the command screen. This will download the driver from ROM. The built-in printer driver is fairly sophisticated, and supports all the extended highlight sequences listed on the View Reference Card.

BASIC ERRORS

Matthew Cooper

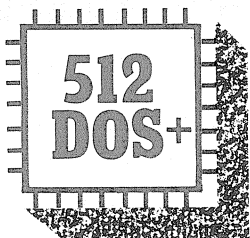
There is a little quirk in Basic that only shows itself if you have a program with a line 0 in it. As you will know, normally when a program generates an error, Basic prints a message such as:

Syntax error at line 1234
However, if an error occurs at line 0 in a program, the message is not:

Syntax error at line 0
but just:

Syntax error

This is because Basic looks at the line number where the error occurred, and if it is zero, assumes that the error was in a direct mode command and doesn't bother to print the line number. While this should cause no problems, it can be confusing if you do not realise what is happening. B



*Robin Burton
presents another
collection of
hints for users of
the 512 co-
processor.*

In this month's forum we will expand on a couple of points from the example batch file given last time, and then look at a feature of the 512 that many owners are quite unaware of.

COLOUR

It seems that there is quite a lot of confusion among new users of the 512, as to the use of colour displays. The 512 emulates a CGA (Colour Graphics Adaptor), and a CGA display in 80 column mode can use four colours, so you might hope the 512 will do this too. Unfortunately this is not so: in 80 column mode the 512 displays only two colours.

The 512's CGA emulation is limited to ensuring that the four colour output from an application is translated to two colours, while still producing a legible display. For the reason behind this look no further than the BBC's two 80 column screen modes (mode 0 and mode 3). Both of these are two colours only.

The colours can be any of the BBC's 16, but only two at a time, one for the foreground, and one for the background. In practice, only the non-flashing colours in the range 0 to 7 are of any real use.

Even if your software includes four colours in its options, and you have seen it use four colours and 80 columns on a PC, the 512 still displays only two. The normal choice of colours is white text on a black background, although this can be changed by using the 'colour' command, which performs a VDU 19 to change the screen colours.

The 'colour' command takes two parameters: the first indicates whether the colour to be changed is the foreground colour (1) or the

background colour (0), while the second is the number of the colour to be used. The second parameter can be in the range 0 to 15, and is the physical colour number. These are listed in the Beeb's User Guide. The 'colour' command works by simply sending the necessary VDU codes to the Beeb to bring about the colour change. For example, to display green text on a blue background, the two commands required are:

```
colour 0 4 - Change the background to  
             blue  
colour 1 2 - Change the foreground  
             (text) to green.
```

It does not matter what the colours are to begin with, they will be changed regardless. Another thing to note is the use of spaces, rather than commas, to separate parameters. This is standard practice with DOS+ commands. To set the colours for GEM, use 2 and 3 instead of 0 and 1 for background and foreground respectively.

If you are using GEM, it is possible to have a four colour display, by installing the four-colour screen driver supplied on the miscellaneous disc (number four). This is done using the GEM set-up program and is described in the 512 User Guide. The main drawback of this four-colour driver is that there is a straight trade-off of screen resolution against colour. That is, you double the colours, but you halve resolution. Although 80 column text can be displayed, it isn't suitable for word processing, even with GEM Write, owing to the poor character resolution.

The 'colour' command, when used with the four colour option, is similar except that there is no foreground or background as such. Each of the four colours is referred to by a number between four and seven. For example, to change the first colour (number four) to cyan (BBC colour six), the command is:

```
colour 4 6
```

If you later wanted to change colour four, currently cyan, to blue, the colour number does not change, so the command would be:

```
colour 4 4
```

THE RAM DISC

Next we will take a look at the RAM disc, which is an area of memory set up to behave as if it were another disc drive. I suggested that this could be used with batch files to avoid disc swapping. In fact it can be useful with many programs, with or without batch files. Some implementations of DOS have a (semi) permanently configured default RAM disc which appears automatically when the system's loaded. This is not possible on the 512, but given its limited memory this is just as well.

Memory is at a premium on the 512. As it is, many programs will not run with less than 640K of memory. You therefore have to be very careful about how much memory you allocate to a RAM disc, as this will reduce the amount of memory available to programs. Some programs need all, or most, of the available RAM, for loading, even if they can cope with less memory when running. These applications will fail with a RAM disc installed, so you must experiment with your own jobs. What works for one may very well not suit another.

While we're on this topic, if you wonder why some programs run quite happily in a 512K MS-DOS machine but won't run in a Master 512, the reason is that DOS+ takes about 90K more RAM than MS-DOS.

To set up a RAM disc is simplicity itself; even the User Guide explains this well enough. It is what it doesn't tell you that matters. The command is 'memdisk', followed by the number of kilobytes that you want to allocate. For example, to create a 20K RAM disc you would enter:

```
memdisk 20
```

If you use a RAM disc you can include its set-up commands in your startup batch file. Include the command to create it, together with suitable copy statements to put into it the often needed transient command software, like 'fset', 'chkdsk' and your batch files. When these are called, you won't have to re-insert the systems disc, nor do you need a copy of them on every data or applications disc. Try these commands

with the systems disc in drive A - you will soon get the idea:

```
backg
memdisk 60
copy a:fset.cmd m:
copy a:chkdsk.cmd m:
dir m:
backg
```

The two 'backg' commands show the memory reduction caused by the RAM disc. The bad news is that memdisk is a program, and it uses memory. As you will see from your screen, the total for a 60K RAM disc is about 150k. The 90K overhead does not vary, so even a 20K RAM disc needs 110K. Bear this in mind when you decide on the suitability of a RAM disc for your particular job.

Important points to note about a RAM disc are that after it is created you are stuck with it until the system is re-booted. You cannot remove it or change its size even if it is empty. This is a real nuisance, and definitely belongs on the bugs/omissions list, although in fairness this is peculiar to neither the 512, nor even DOS+.

Another point is, no matter what size you allocate, the result is always an even number. If you need a 31K RAM disc you must allocate 32K. An attempt to create it as 31K will be rounded down to 30K by the system, which is perhaps not enough if you are being careful with the size. The final point is that your RAM disc is always drive M.

It is not all bad news, however. With the right applications a RAM disc can greatly speed up operations that call for the loading of transient commands, like 'fset'. They only take a few seconds really, but it seems like an age while you wait. A good many applications do not need all the RAM, and some will adjust to use what's left (within reason) after the other allocations are taken.

For example I use PC-Write, which is quite happy to co-reside with a 100K RAM disc while editing a 60K document in memory, adequate for most purposes. If the RAM disc allocation is 128K, PC Write adjusts by reducing the

maximum document size down to about 42K, and so on.

Unless you have a hard disc, a RAM disc can save you time, but remember that it is not a real disc. A power cut or any machine failure is fatal, as the contents are lost. Use it for temporary storage, but NOT for critical data files.

MONITORING GOINGS-ON

We will round off with a quick look at the monitor built into the DOS+ boot chip, and the subject of segmented memory. This is the program that gives the * prompt, seen when the 512 is switched on but not booted, or when BREAK is pressed before switching off the 512 or entering the 'NOTUBE' command. This is really only for interest, as it is a tool for patching or debugging. Far better general user development tools exist, even among the free ones in the public domain.

To see the full range of the monitor's commands, from the * prompt type:

H.MON

and press return. It is quite limited, but it does provide the ability to access the BBC's star commands as well as the opportunity of getting into the 512's memory directly.

The first command, 'd', allows an area of memory to be dumped, in hex and ASCII.

The command shown as 'dos' simply enters the boot routine to load DOS+.

'f' allows a specified area of memory to be filled with a given byte or word value.

'go' is for debugging specific areas of programs, by supplying the address from which execution should start.

'mon' enters the monitor. This allows the monitor to be re-entered from within DOS+ by typing:

star mon

's' means set. This is the 512 memory editor. You can move around using the cursor keys, and directly key in values as required.

'sr' is a memory search routine allowing a given string to be located in the 512, again used for debugging and testing.

Finally 'tfer', the most interesting command, transfers memory across the Tube in either direction. The parameters are the address in the I-O processor (the BBC) the address in the 512, the length of data to move and R (read from the BBC), or W (write to the BBC) for the direction of transfer. If you feel adventurous try transferring an area of (used) 512 memory to the BBC, then *SAVEing it to disc, when you find 'putfile' or 'move' too boring.

Even more useful, you can call 'tfer' from Basic. Using a simple memory management procedure and 'OSCLI', you could archive up to half a megabyte of BBC Basic data in the 512. It gives a whole new meaning to 'sideways' expansion, doesn't it?

SEGMENTED MEMORY

You will notice that in the help list for the 512 Monitor, addresses are listed as <segment:offset>. What this means is that the addresses should be specified as two values, separated by a colon, rather than the normal single value. The reason for this is that the 80186 can address up to 1 Mbyte of memory. This requires a twenty-bit address, but all the registers used in the 80186 are only sixteen bits wide. To allow the full address range to be accessed using the value from a single register, addresses are made up of two parts - a segment number and an address offset. The processor takes these two parts and forms a twenty bit address by multiplying the segment number by 16 and adding the address offset.

When we specify addresses for use by the Monitor, the segment number is given, followed by a colon, and finally the address offset. So, for example, an address of &12345 could be specified as 1234:5, which means the 5th byte in segment &1234. There are however, other ways of specifying the same address. For example, 1233:15, because &1233*16+&15 is &12345.

B



POSTBAG



POSTBAG

MASTERING DOMESTIC ACCOUNTS

Some years ago BEEBUG published a Domestic Accounts program, and later this was included in the members' pack for a time. Due to changes in the OSBYTE &87 call, the screen dump section of this program does not work correctly on Master series machines. The following changes will correct this:

```
5100 DATA F4FFE0FFF004E000
5110 DATA D00A208357A98720
5120 DATA F4FFA920A90120EE
5330 IF T%<>23804 PRINT"C
checksum error.":END
```

I hope this is of interest.

T.D.Tuddenham

This program has remained a firm favourite with BEEBUG members since it was first published (Vol.2 No.10 & Vol.3 No.6). Several members have asked for help with running this program on a Master, and we are therefore pleased to pass on Mr Tuddenham's update.

OF SHOES AND SHIPS AND SEALING WAX

You asked for comments as to the future direction of BEEBUG. I would be happy to see programs in other languages, simply out of interest. I still have trouble enough with Basic.

I have recently been given an Archimedes for my job as an advisory teacher involved with I.T. (can't wait for a state-of-the-art word processor and

desk-top publishing package), and I would find it very useful if programs published in BEEBUG were marked so as to indicate whether they would run on an Arc unmodified, or under the emulator. I still run many BEEBUG visuals and use programs such as Zoom (BEEBUG Vol.4 No.2), and it would be useful to know if anything new could be used.

As a previous network manager, it would also be useful to see the occasional network comment, and to know whether programs ran also on these systems.

Martyn Wilson

Some other readers have commented similarly about marking programs from BEEBUG for their suitability for running on an Arc. Many programs will run with little or no modification, and often benefit from the increased speed of the Archimedes, though there are some significant differences (see article by Chris Drage in Vol.6 No.10), and clearly any which use the 6502 assembler would only run under the emulator. We hope to experiment with the testing and marking of BEEBUG programs in this way very shortly. Further comment would be welcome.

Readers may also be interested to know that BEEBUG is an Econet approved dealer and runs an Econet network with various BBC micros and an Archimedes connected. This system is now used for all editorial work.

CREDIT WHERE CREDIT IS DUE

I think it is about time you gave more credit to some of your own (Beebugsoft) programs, especially when comparing them with other products of a similar nature in reviews. In particular I refer to Masterfile II which I have used for many years on my BBC, and then on my Master. In my view it is by far the best database going today.

E.A.Allchin

Mr Allchin's letter raises a point which has always caused us some concern. Whilst we accept that members clearly wish to know about any products produced by ourselves, it is also our view that any attempt to review our own products, particularly in comparison with other similar ones, would at the very least undermine the perceived objectivity of our reviews. As a result, other suppliers might no longer be prepared to submit their own products for review, and members might well have less regard for what we say. We have therefore followed a policy for some time now of not reviewing our own products, although we have relaxed this stance with regards to BEEBUG Surveys (starting this month) of existing rather than new products. We hope members will appreciate our position in this in the interests of providing independent and objective reviews of all products which we do feature in BEEBUG. **B**

BEEBUG MEMBERSHIP

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

BEEBUG SUBSCRIPTION RATES

£ 7.50	6 months (5 issues) UK only	£23.00
£14.50	1 year (10 issues) UK, BFPO, Ch.1	£33.00
£20.00	Rest of Europe & Eire	£40.00
£25.00	Middle East	£44.00
£27.00	Americas & Africa	£48.00
£29.00	Elsewhere	

BEEBUG & RISC USER

BACK ISSUE PRICES (per issue)

Volume	Magazine	Cassette	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.50	£3.50	-
3	£0.70	£2.00	£4.00	£4.50
4	£0.90	£2.50	£4.50	£4.75
5	£1.20	£3.00	£4.75	£4.75
6	£1.30	£3.50		
7	£1.30			

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

FURTHER DISCOUNTS

We will allow you a further discount:

Five or more:	deduct £0.50 from total
Ten or more:	deduct £1.50 from total
Twenty or more:	deduct £3.50 from total
Thirty or more:	deduct £5.00 from total
Forty or more:	deduct £7.00 from total

POST AND PACKING

Please add the cost of p&p:

Destination	First Item	Second Item
UK, BFPO + Ch.1	40p	20p
Europe + Eire	75p	45p
Elsewhere	£2	85p

BEEBUG
Dolphin Place, Holywell Hill, St.Albans, Herts AL1 1EX
Tel. St.Albans (0727) 40303, FAX: (0727) 60263

Manned Mon-Fri 9am-5pm

(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: David Spencer
Technical Assistant: Lance Allison
Production Assistant: Yolanda Turuelo
Membership secretary: Mandy Mileham
Editorial Consultant: Lee Calcraft
Managing Editor: Sheridan Williams

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Limited.

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

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

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

BEEBUG Ltd (c) 1988

Printed by Head Office Design (0782) 717161 ISSN - 0263 - 7561

Magazine Disc/Cassette

AUG/SEPT 1988 DISC/CASSETTE CONTENTS

CROSSWORD EDITOR - a highly useful program for all crossword fans for both designing and solving crosswords, with optional access to SpellMaster for even more assistance.

RUNNING A TEMPERATURE - two programs, one to calibrate and one to record from the simple temperature probe described this month.

MATHEMATICAL WORMS - explore this fascinating world in the mould of Life and Mandelbrot.

BEEBUG MINIWIMP (PART 3) - a demonstration of the use of the BEEBUG MiniWimp for pull-down menus.

VIEWING FOREIGN PARTS - word processing with View in French, German, Spanish, Greek and Turkish.

MATRICES IN BASIC (PART 2) - additional code to implement two further matrix operations in BBC Basic.

FIRST COURSE

JUST SCROLLING - seven separate procedures for scrolling text in a variety of fancy ways, all packaged up in one complete demonstration program.

FILE HANDLING FOR ALL (Part 4) - a complete working database program to demonstrate the use of the procedures listed in the magazine.

USING ASSEMBLER (Part 2) - three assembler programs showing the development of a generalised plotting routine.

BEEBUG WORKSHOP - how to use linked lists to store data in an ordered sequence.

DUAL SCREEN PROGRAM DISPLAY - display and compare two Basic programs on the screen at the same time.

MAGSCAN - bibliography for this issue (Vol.7 No.4).

All this for £3 (cassette), £4.75 (5" & 3.5" disc) + 50p p&p.

Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

SUBSCRIPTION RATES
6 months (5 issues)
12 months (10 issues)

5" Disc
£25.50
£50.00

UK ONLY
3.5" Disc
£25.50
£50.00

Cassette
£17.00
£33.00

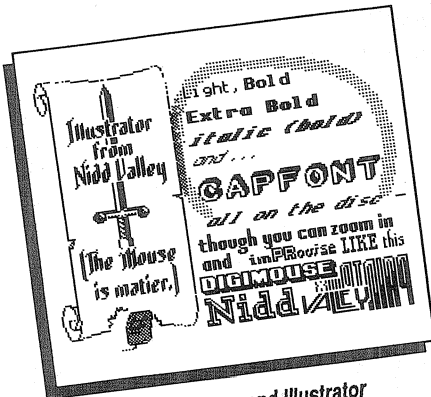
5" Disc
£30.00
£56.00

OVERSEAS
3.5" Disc
£30.00
£56.00

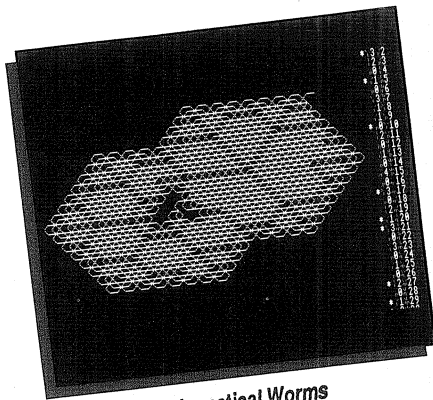
Cassette
£20.00
£39.00

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

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:
BEEBUG, Dolphin Place, Holywell Hill, St. Albans, Herts. AL1 1EX.



Paintbox and Illustrator



Mathematical Worms

5% Discount On ALL Archimedes From BEEBUG In Addition To...

The **Archimedes**
Specialists

1 0% FINANCE

For a limited period we are able to offer 0% APR finance over 9 months on the purchase of any Archimedes. You pay no interest at all. This is a **brand new** scheme only available from BEEBUG. The deposit and repayments are shown below.

Deposit 9 Payments			Deposit 9 Payments		
A305			A310		
A305 Base	£79.66	£76.00	A310 Base	£93.24	£91.00
A305 Mono	£91.21	£82.00	A310 Mono	£104.79	£97.00
A305 Colour	£104.01	£100.00	A310 Colour	£117.59	£115.00
A310M			A440		
A310M Base	£104.79	£97.00	A440 Base	£278.93	£276.00
A310M Mono	£107.34	£104.00	A440 Mono	£290.48	£282.00
A310M Colour	£129.14	£121.00	A440 Colour	£303.28	£300.00

4 FREE PC EMULATOR AND 1st WORD PLUS

Purchase your Archimedes by Cheque, Access, Visa, Official Order or 11.5% finance and we will supply you, absolutely free, 10 3.5" discs, a lockable disc storage box, printer lead and the latest version of The PC Emulator from Acorn. Additionally if you are purchasing a 440 system you will receive 1st Word Plus.

Prices Including VAT		
A305 Base	£763.66	Mono £829.21
A310 Base	£912.24	Mono £977.79
A440 Base	£2762.93	Mono £2828.48
		Colour £1004.01
		Colour £1152.59
		Colour £3003.28

2 TRADE IN YOUR OLD BBC, MASTER OR COMPACT FOR AN ARCHIMEDES

We will be pleased to accept your old computer (in working condition) as part exchange towards the purchase of an Archimedes. (If you use the finance scheme this will replace your initial deposit on a 305/310, so you pay nothing now). Allowances are as follows:

BBC Issue 4 No DFS	£125	BBC Issue 4 DFS	£175
BBC Issue 7 No DFS	£175	BBC Issue 7 DFS (Or B+)	£225
Master 128	£250	Compact Base System	£215

Please phone for allowances on other Compact and Master systems.

3 EXPORT

Although unable to offer finance to overseas customers, we can offer an efficient export service with delivery to your door. Please write for a quotation.

TO FIND OUT MORE PHONE OR WRITE NOW. TEL: 0727 40303

We offer a complete service, including Advice, Technical Support, Showroom, Mail Order and Repairs. Our showroom in St. Albans stocks everything available for the Archimedes. Call in for a demonstration.

5 11.5% FINANCE OVER 12 TO 36 MONTHS

As a Licensed Credit Broker we are able to offer finance on the purchase of any equipment, including the Archimedes. You still benefit from the free PC Emulator, discs, disc box and printer lead. (Typical APR 23% on the purchase of a 310 Colour system over 36 months.

Deposit £152.59 36 payments of £37.36).

6 DISCOUNTS FOR EDUCATION

We are able to offer attractive discounts to Education Authorities, Schools, Colleges and Health Authorities. Please write with your requirements for a quotation.

These discounts are available to you as a member of RISC User, THE leading magazine dedicated solely to Archimedes. So if you are not yet a member, join now for only £14.50 for a full year.

Please indicate your requirements below.

Subscriptions to Risc User (£14.50 UK) ☐ Information Pack and Catalogue ☐ 0% Finance Form for 305/310/310M/440 Base/Mono/Colour ☐ 12-36 Months Finance Form for 305/310/310M/440 Base/Mono/Colour ☐ Trade In BBC/Master/Compact ☐ Purchase 305/310/440 Base/Mono/Colour ☐ UK Courier Delivery £700. Overseas please ask for a quotation.

I enclose a cheque value £.....

Please debit my Access/Visa/Connect Card No.....

Expiry..... with £.....

Name.....

Address.....

Signature.....

Beebug, Dolphin Place, Holywell Hill, St. Albans, Herts AL1 1EX Tel: 0727 40303

BEEBUG - The Archimedes Specialists