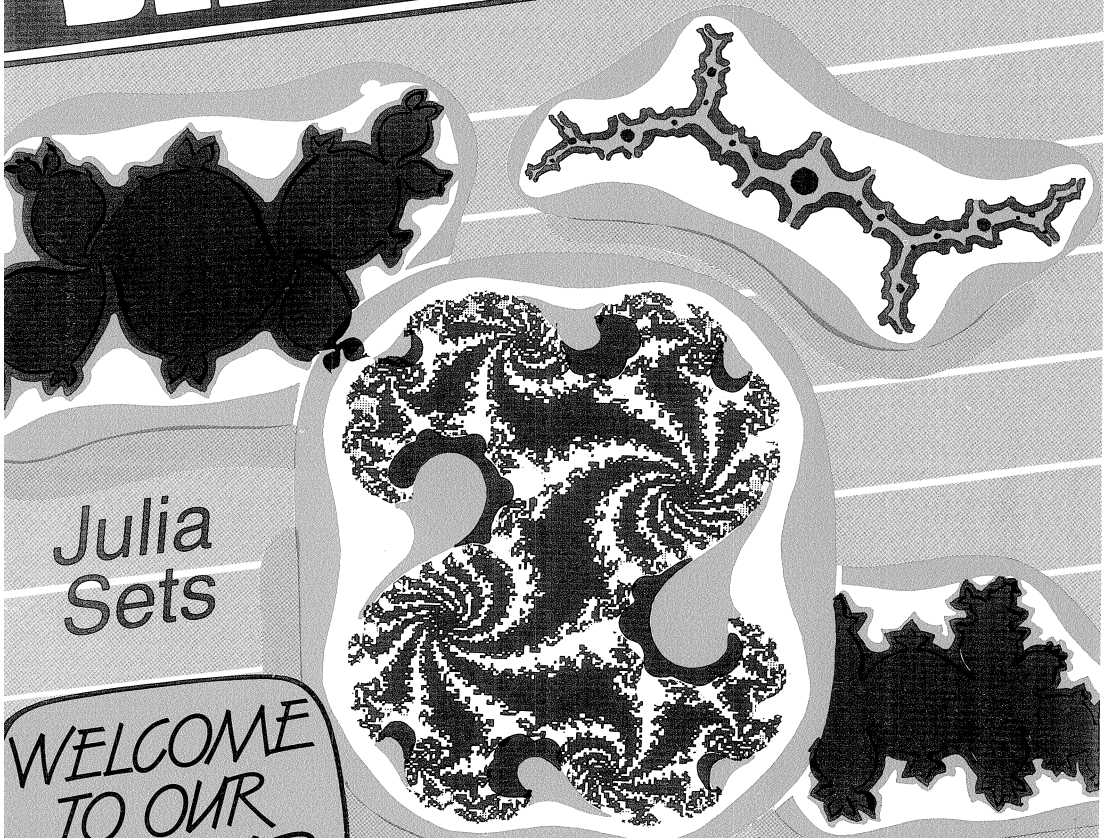
Vol.8 No.1 May 1989

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



Julia Sets

WELCOME TO OUR 8th YEAR

- TRANSPARENT SWR LOADER
- SHARE INVESTOR
- BOUNCER BALL GAME
- SPEED & SOUND CONTROL

BEEBUG Vol.8 No.1 May 1989

FEATURES

REVIEWS

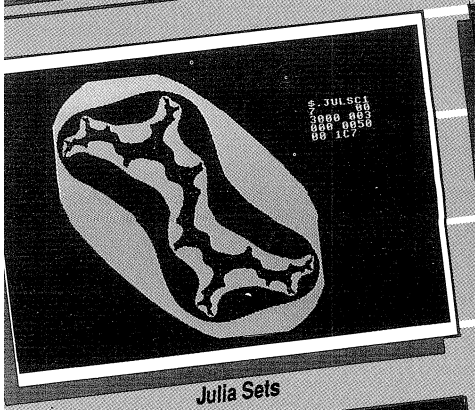
REGULAR ITEMS

HINTS & TIPS

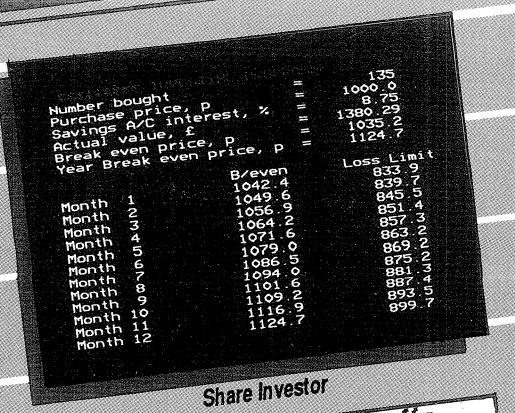
PROGRAM INFORMATION

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

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

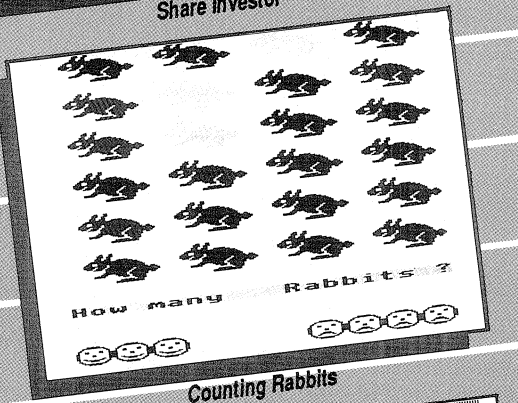


Julia Sets

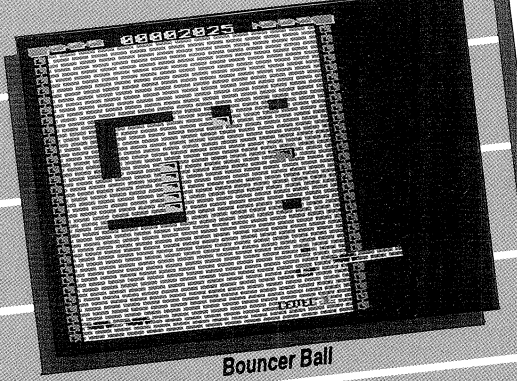


Share Investor

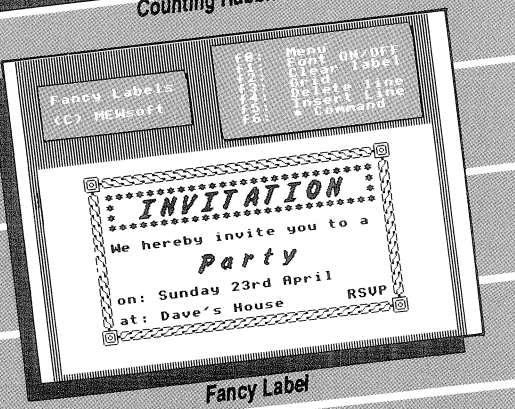
Termin Review



Counting Rabbits



Bouncer Ball



Fancy Label

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

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

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

BEEBUG EIGHT YEARS ON

This issue marks the start of the eighth year of publication of BEEBUG. That amounts to 70 magazines containing some 750 quality programs, 700 hints and tips, and over 600 reviews of software, hardware and books. We have also published many series of articles addressing the needs of particular groups of users, our First Course for newcomers to computing, the Comms spot for those interested in communications, series on assembler language programming and file handling, BEEBUG Education, articles and information on the Master series of micros, and more recently the 512 Forum for users of this second processor.

Not only that, but membership of BEEBUG brings other important benefits, particularly free access to our Technical Team for advice and help on all manner of queries and problems relating to your micro, and also our friendly shop and mail order service with its discounts for members.

We feel that this is an impressive record, and one on which we can continue to build. Despite the advent of other newer machines from Acorn, the BBC micro and the Master series still account for the vast majority of all Acorn micros in use, and these machines will provide faithful service to their owners for many years to come. BEEBUG is still the only magazine which continues to be exclusively devoted to BBC micro and Master users. As a user, we will do our best to ensure that the magazine continues reflects your needs and interests in the future.

VOLUME 7 INDEX

As usual with the start of a new volume of the magazine we have enclosed with this issue a comprehensive index to the contents of all magazines in volume 7.

BBC ACORN USER SHOW

As we announced, there will be an Acorn User Show this year, though with a slightly changed title reflecting the BBC's purchase of Redwood, the publishers of Acorn User. The show takes place from 21st to 23rd July at Alexandra Palace in north London, and (as far as we know at present) will be the only show this year for users of Acorn machines. We will have a stand at the show for both BEEBUG and RISC User (stand 55), so mark these important dates in your diary now. Note that on the first day (Friday) the show will be open from 3pm to 9pm to make it more accessible.

This month's telesoftware password is *commando*.

LOOKING AHEAD WITH BEEBUG

The following features are likely to be included in the June issue of BEEBUG:

Applications & Utilities:

- Four Channel Temperature Probe
- Mathematical Transformations
- Abbreviation Substitution
- Disassembler
- Share Analysis Part 2
- File Handling for All
- Fast DFS Backup
- Foreign Language Vocabulary Tester

Reviews:

- Music 5000 Junior
- Ceefax Weather Pictures
- DOS+ Utilities

Plus News, Postbag, Hints and more.

RISC USER

RISC User is the largest circulation magazine devoted entirely to the Acorn Archimedes range. It is available on subscription to all BEEBUG members at a substantially reduced rate (see page 66).

We expect the June issue to include:

- Desktop Auto-Install
- Multi-task Notepad Part 3
- Procedure Libraries
- Converting Text Files to 1st Word Plus
- RISC OS Scaled Text
- RISC OS Printer Drivers
- 24 Pin Printer Dump
- New Series on Basic

Reviews:

- Acorn's New Micro
- System Delta Plus Version 2
- Watford Electronics Hard Disc

and more.

RISC User is the ideal magazine to keep up to date with the Archimedes scene in every respect, and is particularly useful if you are contemplating the purchase of an Archimedes in the near future.

*The latest details of the contents and distribution of both magazines are contained in the BEEBUG area of Micronet. Just type *BEEBUG# when on-line.*

News News News News News News

ARCHIMEDES 400 SERIES

In last month's BEEBUG news section we exclusively announced the new Archimedes 410/1 and 420/1. Acorn has now also announced the 440/1. This machine will have a similar specification to the existing 440, but features a new 50Mbyte hard disc. Additionally, all the 400/1 series are an average of 10% faster than the current Archimedes machines. This is achieved by a new version of MEMC, the memory controller chip, which optimises RAM access cycles.

The prices given last month for the 400/1 series were preliminary figures. The revised retail prices for a base system (no monitor) are:

410/1 £1378.85
420/1 £1953.85
440/1 £2873.85

All prices are inclusive of VAT. You will notice that the 440/1 is marginally cheaper than the current 440, even though it offers increased mass storage.

NEW COMPUTER

Acorn are set to launch their latest RISC based computer, believed to be a budget priced version of the Archimedes. Full details will be released in mid-May, and reported in next month's BEEBUG. If you can't wait until then, details will also appear in the June issue of our sister magazine, RISC User.

ACORN BACK IN THE BLACK

Acorn's financial situation at the end of 1988 was a great improvement on the previous year's figures. The company made a profit of £1.6 million against last year's loss of £2.4 million. This was the result of an increase in turnover from £36.1 million to £39.2 million. The earning per share is 1.7 pence, although no dividend payment is being recommended.

Acorn's managing director, Harvey Coleman, said the profits were the result of a year which included the 300,000th Master sold, the announcement of RISC OS and the R140 UNIX workstation. Coleman went on to say that Acorn put back 10% of all turnover into research and development, and that they had just received the first prototypes of the ARM 3 chip - the 300,000 transistor RISC chip which will form the heart of the Acorn computers of the '90s.

AMATEUR WEATHER FORECASTING

BBC Telesoftware has started broadcasting pictures received from the weather satellite *Meteosat*. These pictures, which show the weather trends over the earth's surface, are transmitted in a standard telesoftware format which can be decoded into actual pictures using a program which can also be downloaded. The service is totally free of charge, and all you will need is a model B, B+, Master or Archimedes, and a suitable teletext adaptor. For further details contact BBC Telesoftware, CEEFAX, BBC TV Centre, Wood Lane, London W12 7RJ.

PRICEY PRINTER

If you fancy a printer with a difference (and a large price tag), then one to try is the Qume *Crystal Print Publisher*. This offers similar performance to a laser printer, but in fact uses a liquid crystal shutter. It supports a full implementation of *PostScript* (see article in last month's BEEBUG), and at £3449 (inc. VAT) is the cheapest *PostScript* printer available. Further details from Qume, Qume House, Parkway, Newbury, Berks RG13 1EE.

REPEAT PERFORMANCE

Play it again Sam 8 is the latest of Superior Software's games compilation. The package contains four classic games - Winter Olympiad '88, Quest (an arcade-adventure), Repton around the world in 40 screens, and Mr. Wiz. *Play it again Sam 8* costs £9.95 on tape, £11.95 on 5.25" disc, and £14.95 on 3.5" disc (all inc. VAT), and can be obtained from Acorn dealers, or direct from Superior Software, Regent House, Skinner Lane, Leeds LS7 1AX, tel. (0532) 459453.

BEEB CIRCUIT DESIGN

CIRDLIN is a CAD program for the model B and Master which is designed to help CDT students. The system is primarily aimed at the drawing of circuit diagrams, and includes many built-in circuit icons, although the CAD facilities can be used for any application. *CIRDLIN* was developed as part of the Institute of Electrical Engineer's UNCLE scheme which is designed to increase awareness of the electronics industry among young people. The package costs £15.00 on a 5.25" disc, or £17.50 for a 3.5" version, and can be obtained from Peter Helsdon, 32 Burns Crescent, Chelmsford, Essex CM20 0TS. B

Julia Sets

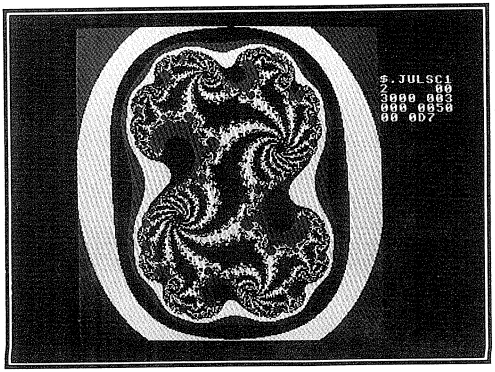
John Greening opens up another fascinating visual area for exploration in his program for displaying Julia sets. Puzzled? Well read on.

It would be surprising by now if you had not heard of the Mandelbrot Set, as almost all computer magazines have published programs for reproducing sections of it. But if you have not heard of Julia sets there is much more excuse. Julia is not the name of a lady but of one French mathematician (the other was Fatou) who, around the time of the first World War, investigated iterative calculations in the complex plane - without the benefit of the modern computers available to Benoit Mandelbrot.

Both the Mandelbrot set and Julia sets (the plural is important) are obtained from repeated calculations of the form:

z=z^2+c

where both z and c are complex numbers. For the Mandelbrot set z is set equal to zero and c varied systematically over its real and imaginary parts. To get a Julia set, c is fixed and z is varied systematically, but for every c a different Julia set is obtained. For more information see *Scientific American* Nov. 1987 p.118, or *The Science of Fractal Images*, Springer Verlag, 1988.



For every point in and around the Mandelbrot set there is a so-called Julia set. If the point lies within the Mandelbrot set the Julia set is continuous; if it lies outside, the Julia set is fragmented to a greater or lesser extent; and if it is on the edge of the Mandelbrot set the Julia set is a line. Examples of all of these are given in this article.

The program is straightforward to enter, and to run, but note the later comments about the necessary modifications for different versions of Basic. When the program is run it prompts you to enter values for g and h. These are the x(real) and y(imaginary) co-ordinates of the point in the Mandelbrot plane for which the Julia set is to be plotted. Some suggestions for these values are given in table 1.

No.	g (real)	h (imag.)
1	.32	.043
2	-.481762	-.531657
3	.27334	.00742
4	-.39054	-.58679
5	0	1.0
6	-.11	.6557
7	-.15652	1.03225
8	-.74543	.11301
9	-.11	.67
10	-.194	.6557
11	.31	.04
12	-.12	.74
13	-1.25	0
14	.11031	-.67037

Table 1. Suggested values for g and h

Most interesting Julia sets are close to the edge of the Mandelbrot set. You can get a little more detail in the plots by increasing the value of *maxcount%* in line 210, but at the expense of extra plotting time. When you have a good book to read try setting *maxcount%* to 250 and running the first pair of g and h given in table 1.

Programs that plot the Mandelbrot set often require very little modification to plot Julia sets, as similar iterations are used. But as you will know if you have plotted Mandelbrots on the BBC micro, they take a long time. This is not surprising. The mode 1 program given here plots over 65,000 pixels, and for many of them 64 (or

more if you wish) iterations are necessary to determine the colour of the pixel. In consequence, I have provided in the listing an alternative assembler version of the iterative loop to speed things up, but you can do it all in Basic if you wish (see REM statements in the listing).

One other thing that helps to reduce running time is that Julia sets are symmetrical, so that every calculation permits two points to be plotted. Even so you will need patience as the longest program takes one and a half hours, although others are much shorter.

You can speed things up by a factor of four by working in mode 2, but at the cost of poorer resolution. Only minimal changes are necessary to the printed program in order to use mode 2. They are as follows:

```
100 MODE 2
120 VDU28,16,31,19,0
```

In lines 230 and 240 change .011 to .022. In lines 360 to 440 make the (second) GCOL values run from 1 to 7 and then start to repeat. And add the two lines:

```
465 PLOT69,X%,Y%-4
475 PLOT69,-X%,-Y%-4
```

PROGRAM MODIFICATIONS

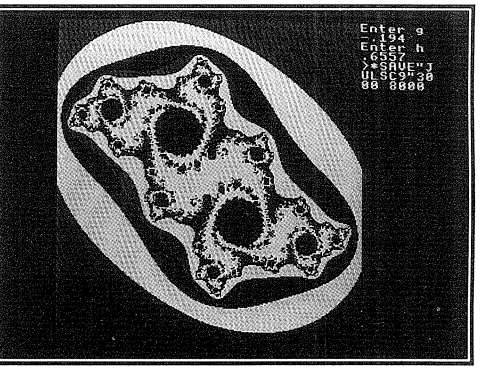
The assembler section utilises floating point procedures in the Basic ROM. The addresses of these differ in different versions of BBC Basic. Those shown in the program at the beginning of the assembler section relate to Basic II, so if you have another version you will need to substitute the appropriate addresses given in table 2.

Subroutine	Basic I	Basic IV
aunp	&A3A6	&A541
amult	&A661	&A6A6
bcopya	&A20F	&A40B
acomp	&AD99	&ACCA
aplusb	&A513	&A692
aplus	&A50E	&A68D
apack	&A37E	&A519
asign	&A1CB	&A1DA

Table 2. Floating point addresses in Basic I and Basic IV

For Basic IV on the Master, you should also ensure that every occurrence of &4B in the listing is replaced by &4A, and every occurrence of &4C by &4B, when you type the program in (but don't change addresses like &4C5). In addition (for Basic IV) lines 1660 and 1670 should be replaced by the two lines:

```
1660 LDA &2E
1670 BMI out
```




```
   10 REM Program Julia Sets
   20 REM Author  John Greening
   30 REM Version 1.1
   40 REM BEEBUG  May 1989
   50 REM Program subject to copyright
   60 REM Program is set up for Basic II
   70 REM See text for changes needed fo
r Basic I & Basic IV
  100 MODE1
  110 VDU29,512;512;
  120 VDU28,32,31,39,0
  130 REM Lines 140 to 170 are needed to
 initialise variables used in the machin
e code version of the program.
  140 DIM mc% 500
  150 r=0:i=0:C%=0
  160 a=0:b=0:d=4:n=0:m=0
  170 x=0:y=0
  180 INPUT"Enter g " g:REM See table
  190 INPUT"Enter h " h:REM See table
  200 REM Change maxcount% if you wish
  210 maxcount%=64
  220 PROCassemble
  230 FOR x=-1.408 TO 0 STEP .011
  240 FOR y=-1.408 TO 1.408 STEP .011
  250 r=x:i=y:C%=0
  260 CALL repeat:REM This is the machin
e code version of the REPEAT loop two li
nes down
```

```
  270 GOTO 350:REM This skips the Basic
REPEAT loop. However, if you put in a li
ne 255GOTO 280 you can do it all in Basi
c!
  280 REPEAT
  290 n=r*r-i*i+g
  300 m=2*r*i+h
  310 C%=C%+1
  320 s=r*r+i*i
  330 r=n:i=m
  340 UNTIL s>4 OR C%>maxcount%-1
  350 IF C%>maxcount%-1 THEN GCOL0,0:GOT
O 450
  360 IF C%>25 GCOL0,3:GOTO 450
  370 IF C%>12 GCOL0,2:GOTO 450
  380 IF C%>8 GCOL0,1:GOTO 450
  390 IF C%>5 GCOL0,3:GOTO 450
  400 IF C%=5 GCOL0,2:GOTO 450
  410 IF C%=4 GCOL0,1:GOTO 450
  420 IF C%=3 GCOL0,3:GOTO 450
  430 IF C%=2 GCOL0,2:GOTO 450
  440 IF C%=1 GCOL0,1
  450 X%=364*x:Y%=364*y
  460 PLOT69,X%,Y%
  470 PLOT69,-X%,-Y%
  480 NEXT:NEXT
  490 REM Insert your screen save or scr
een dump here, e.g.*SAVE"JULIA"3000 8000
  500 END
  510 :
 1000 DEFPROCassemble
 1010 FOR pass%=0 TO 2 STEP 2
 1020 P%=mc%
 1030 [OPT pass%
 1040 .aunp JMP &A3B5
 1050 .amult JMP &A656
 1060 .bcopya JMP &A21E
 1070 .acomp JMP &AD7E
 1080 .aplusb JMP &A505
 1090 .aplus JMP &A500
 1100 .apack JMP &A38D
 1110 .asign JMP &A1DA
 1120 .repeat
 1130 PHP:PHA:TXA
 1140 PHA:TYA:PHA
 1150 .loop
 1160 LDY &4E4:INY:INY:INY:STY &4B
 1170 LDA &4E5:STA &4C
 1180 JSR aunp:JSR amult
 1190 LDY &4C2:INY:INY:INY:STY &4B
 1200 LDA &4C3:STA &4C
 1210 JSR apack:JSR bcopya
 1220 LDY &4D2:INY:INY:INY:STY &4B
 1230 LDA &4D3:STA &4C
 1240 JSR aunp:JSR amult
 1250 LDY &4C4:INY:INY:INY:STY &4B
 1260 LDA &4C5:STA &4C
 1270 JSR apack:JSR acomp
 1280 LDY &4C2:INY:INY:INY:STY &4B
 1290 LDA &4C3:STA &4C
 1300 JSR aplus
 1310 LDY &4CE:INY:INY:INY:STY &4B
 1320 LDA &4CF:STA &4C
 1330 JSR aplus
 1340 LDY &4DC:INY:INY:INY:STY &4B
 1350 LDA &4DD:STA &4C
 1360 JSR apack
 1370 LDY &4E4:INY:INY:INY:STY &4B
 1380 LDA &4E5:STA &4C
 1390 JSR aunp
 1400 LDY &4D2:INY:INY:INY:STY &4B
 1410 LDA &4D3:STA &4C
 1420 JSR amult
 1430 INC &30
 1440 LDY &4D0:INY:INY:INY:STY &4B
 1450 LDA &4D1:STA &4C
 1460 JSR aplus
 1470 LDY &4D2:INY:INY:INY:STY &4B
 1480 LDA &4D3:STA &4C
 1490 JSR apack
 1500 LDY &4DC:INY:INY:INY:STY &4B
 1510 LDA &4DD:STA &4C
 1520 JSR aunp
 1530 LDY &4E4:INY:INY:INY:STY &4B
 1540 LDA &4E5:STA &4C
 1550 JSR apack
 1560 INC &40C
 1570 LDY &4C2:INY:INY:INY:STY &4B
 1580 LDA &4C3:STA &4C
 1590 JSR aunp
 1600 LDY &4C4:INY:INY:INY:STY &4B
 1610 LDA &4C5:STA &4C
 1620 JSR aplus:JSR acomp
 1630 LDY &4C8:INY:INY:INY:STY &4B
 1640 LDA &4C9:STA &4C
 1650 JSR aplus
 1660 JSR asign
 1670 CMP #&1:BNE out
 1680 LDA &40C:CMP #maxcount%:BCC jump
 1690 JMP out
 1700 .jump
 1710 JMP loop
 1720 .out
 1730 PLA:TAY:PLA
 1740 TAX:PLA:PLP
 1750 RTS
 1760 ]
 1770 NEXT pass%
 1780 ENDPROC
```

Share Investor

With the increasing interest in investments in stocks and shares, Graham Marsh describes a program which takes a different approach from most by using the computer to assist in decision making.

INTRODUCTION

With the increase in share ownership, many people should find this program useful, as it helps to draw attention to the significance of fluctuations in share prices, and also to the effects of dealing costs on the actual value of any investment.

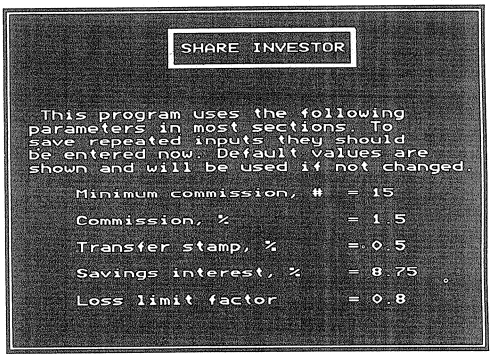
This program is intended for the small investor, where shares are purchased in numbers which often keep the total cost of the transaction within the bottom band of stockbroking fees. A fixed charge is then usually applied, but the program will cater automatically for situations where a percentage fee is used.

There are five aspects to the program which should enable an investor to keep close control over his investments, and help him to reach decisions on when to sell or buy. The program will not make these decisions for him, nor will it guarantee a fortune, but it should help to clarify issues, such as whether or not the money would have been better left in a savings account, how much better it is in shares, and how big a rise in share value is necessary to make a good profit; for instance, a doubling of the share price does not double your money.

THE PROGRAM

By necessity the program is quite lengthy. As a result it is presented in two parts, one this month and one next. This month's listing provides a fully working program, but with two of the menu options missing. These will be listed in the next issue, which will also include information on the program itself for those who may wish to modify it for their own requirements. Do keep to the line numbering when entering the program, and make sure you have saved a copy before running it.

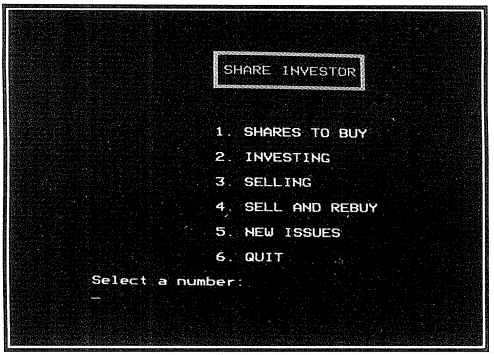
The screen display has been devised to try and make the program self explanatory. On running, you are asked to input certain values which form the basis of subsequent calculations. The meanings of these values should be clear from the screen displays, but a few additional words will not come amiss.



Confirming default parameters

The first two figures are the *minimum* commission which will be applied to any transaction, and a percentage fee if the scale of the transaction is above the minimum. Stamp duty is payable on all purchases of shares, except for new issues, and the current rate is 0.5% as in the default shown. Savings account interest, as a percentage, is the return you would expect on the account where you would normally keep the money if it were not invested in shares. Lastly, the loss limiting factor will help to control the extent of any loss when share values are falling. If no values are input (simply pressing Return), the default values of £15, 1.5%, 0.5%, 8.75% and 0.8 are used. VAT of 15% is assumed. The defaults are set in line 110 if you wish to make permanent changes.

These values are used in the calculations to arrive at the data displayed. There could be slight errors in the final results if there is a change in these values between buying and selling a share, as the input values are used in both calculations, but it should not amount to much.



Main menu

THE MENU

Once the initial data has been determined you are presented with the five menu options, the first three of which are as follows.

1. BUYING OPTION

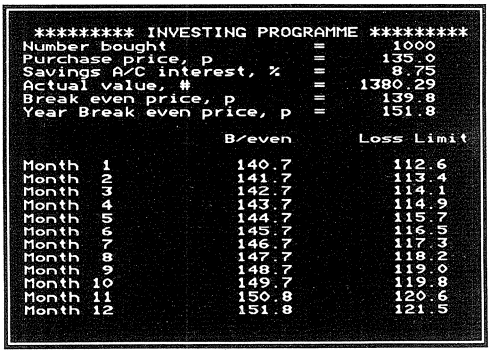
Quite simply this asks how much cash you wish to invest and the price of the share you wish to buy. A number is returned which allows for the broker's fee, transfer stamp and VAT, which is why the number displayed is not that which would be expected from dividing the cash available by the share price.

On 'Continuing', a table is displayed giving an analysis of the investment. Actual value is the cost of the total investment including the expenses of purchase. As can be seen, if an example is run, this significantly affects the share value. The *break even* value is included to illustrate the value a share must reach in the very short term (less than one month), in case an urgent sale is necessary soon after the initial investment with the expectation of recovering the original cash.

Particular importance is placed on the monthly break even value. This is the price the share must reach each month to allow you to sell it and equal the return you would have had from your savings account. Any value greater than this is a bonus. On the other hand, if the market takes a sudden dive (see sell and rebuy next month), it is necessary to make a judgement whether or not to sell at a small loss, hoping to buy in at a still lower price to ultimately recover the loss, and indeed make a profit. This is where the 'Loss limit' column is of use as this gives an immediate indication of what this limiting value should be. The default value of 0.8 sets the loss limit to 20% of the original value.

2. INVESTING OPTION

This is essentially the same as option one, but with a different initial premise. Clearly performance better than a savings account is expected, hence the month by month report which also allows ready comparison with any newspaper listing of the current price. Using the purchase price and the latest price from a newspaper listing in Option 3 will also help to clarify the cash flow.



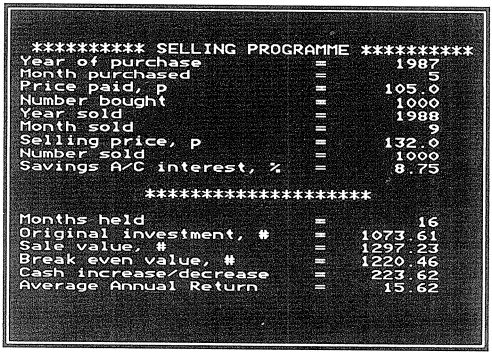
Display from the investment option

3. SELLING OPTION

If sensible decisions on selling are to be made, this section will, hopefully, prove particularly useful. The display is self explanatory with losses being shown in red. A loss in *Annual return* terms is when the value of the transaction is not as great as the sum which would have been realised in a savings account (the break even value for the term of the holding). Hence an annual return of say 5% would show red if this were less than the corresponding savings account interest for the same period, even though there was a cash increase of 5% over the original investment.

When running this section of the program, the current value of the share should be obtained from

your broker for greatest accuracy, since the newspaper price is usually a median value between buying and selling prices, and may in any case be out of date as a paper usually lists values at close of business on the previous day. Often the broker quotes a range of prices based upon how the share has been moving on that day, and this can be significant in terms of profit and loss.



Display from the sell option

NOTE: If the original shares were purchased as new issues, the calculated cash increase will be lower than it should, as the program will assume that the broker's fee and transfer stamp value were paid on the purchase, when in fact there are no such charges for new issues (see next month).

HARD COPY

A printing facility is provided (as an option at the foot of each main display) in case a hard copy of the results is required. Some codes for an Epson printer to set the left margin of the reports and give enlarged title are included (see PROCprint at the end of the listing) but these can be adapted to other printers or deleted altogether.

```
  10 REM Program Share
  20 REM Version B1.4
  30 REM Author  G. Marsh
  40 REM BEEBUG May 1989
  50 REM Program subject to copyright
  60 :
 100 MODE7:ON ERROR PROCerror:END
 110 A=15:B=1.5:C=0.5:D=8.75:E=0.8
 120 FL=0:exit%=FALSE
 130 PROCtitle:PROCintro
 140 A%=A*100:B%=B*100:C%=C*100
 150 D%=D*100:E%=E*100
 160 REPEAT:PROCmenu:UNTIL exit%
 170 END
 180 :
1000 DEF PROCbuy
1010 FL=1:PROCtitle1("SHARES TO BUY")
1020 REPEAT:Ca=FNinput(0,"Cash to inves
t, `",0,28,3):UNTIL Ca>0
1030 REPEAT:Sp1=FNinput(0,"Share price,
 pence",0,28,5):UNTIL Sp1>0
1040 Nr1=INT(FNnumber):Cost=FNcost(Nr1,
Sp1,1)
1050 PRINTTAB(0,12)"Number to buy: ";
1060 PRINTTAB(20,12)Nr1;
1070 PRINT" shares":PROCcont2
1080 CLS:PROCreport1:PROCcont1
1090 ENDPROC
1100 :
1110 DEF PROCinvest
1120 FL=2:PROCtitle1("INVESTING PROGRAM
ME")
1130 REPEAT:Sp1=FNinput(0,"Purchase pri
ce, pence",0,28,3):UNTIL Sp1>0
1140 REPEAT:Nr1=FNinput(0,"Number bough
t",0,28,5):UNTIL Nr1>0
1150 Cost=FNcost(Nr1,Sp1,1)
1160 CLS:PROCreport1:PROCcont1
1170 ENDPROC
1180 :
1190 DEF PROCsell
1200 FL=3:PROCtitle1("SELLING PROGRAMME
")
1210 REPEAT:Yr1=FNinput(0,"Year bought
(1900 - 1999)",0,28,3)
1220 UNTIL Yr1>1900 AND Yr1<2000
1230 REPEAT:M1=FNinput(0,"Month bought
(1 - 12)",0,28,4):UNTIL M1>0 AND M1<13
1240 REPEAT:Sp1=FNinput(0,"Price paid,
pence",0,28,5):UNTIL Sp1>0
1250 REPEAT:Nr1=FNinput(0,"Number bough
t",0,28,6):UNTIL Nr1>0
1260 REPEAT:Yr2=FNinput(0,"Year sold ("
+STR$(Yr1)+" - 1999)",0,28,7)
1270 UNTIL Yr2>1900 AND Yr2<2000 AND Yr
2>=Yr1
1280 REPEAT:M2=FNinput(0,"Month sold (1
 - 12)",0,28,8)
1290 UNTIL M2>0 AND M2<13 AND (NOT(Yr2=
Yr1) OR (M2>=M1))
1300 REPEAT:Sp2=FNinput(0,"Selling pric
e, pence",0,28,9):UNTIL Sp2>0
1310 REPEAT:Nr2=FNinput(0,"Number sold"
,0,28,10):UNTIL Nr2>0
1320 Cost=FNcost(Nr1,Sp1,1):Mo=FNmonths
1330 CLS:PROCreport3:PROCcont1
```

```
 1340 ENDPROC
 1350 :
 1510 REM *** FUNCTION/PROCEDURE LIST **
*
 1520 DEF PROCintro
 1530 LOCAL X,row:row=3
 1540 PRINTTAB(1,row)CHR$131;"This progr
am uses the following"
 1550 PRINT CHR$131;"parameters in most
sections. To "
 1560 PRINT CHR$131;"save repeated input
s they should"
 1570 PRINT CHR$131;"be entered now. Def
ault values are":PRINT CHR$131;"shown an
d will be used if not changed."
 1580 X=FNinput(A,"Minimum commission, `
",5,30,row+6):IF X>0 THEN A=X
 1590 X=FNinput(B,"Commission, %",5,30,r
ow+8)>0:IF X>0 THEN B=X
 1600 X=FNinput(C,"Transfer stamp, %",5,
30,row+10):IF X>0 THEN C=X
 1610 X=FNinput(D,"Savings interest, %",
5,30,row+12):IF X>0 THEN D=X
 1620 X=FNinput(E,"Loss limit factor",5,
30,row+14):IF X>0 THEN E=X
 1630 ENDPROC
 1640 :
 1650 DEF FNinput(deflt,msg$,x1,x2,row)
 1660 PRINTTAB(x1,row)msg$;TAB(x2-2,row)
"= ";
 1670 PRINTTAB(x2,row)SPC8;TAB(x2,row)ST
R$(deflt):INPUTTAB(x2,row)X
 1680 =X
 1690 :
 1700 DEF PROCtitle:VDU26:CLS
 1710 LOCAL J,L,T$:T$="SHARE INVESTOR"
 1720 L=LEN(T$):L1=(39-L)DIV2
 1730 VDU31,L1,0,146,183
 1740 FOR J=1 TO L:VDU163:NEXT
 1750 VDU235:VDU31,L1,1,146,181
 1760 PRINT T$;:VDU234,31,L1,2,146,117
 1770 FOR J=1 TO L:VDU240:NEXT:VDU250
 1780 VDU28,0,24,39,3
 1790 ENDPROC
 1800 :
 1810 DEF PROCtitle1(msg$):VDU26:CLS
 1820 LOCAL L:L=LEN(msg$):L=(37-L)DIV2
 1830 PRINTTAB(0,0);CHR$134;STRING$(L,"*
");" ";msg$;" ";STRING$(L,"*")
 1840 VDU28,0,24,39,1
 1850 ENDPROC
 1860 :
 1870 DEF PROCmenu
 1880 LOCAL row:row=3:CLS
 1890 PRINTTAB(13,row);"1. BUYING"
 1900 PRINTTAB(13,row+2);"2. INVESTING"
 1910 PRINTTAB(13,row+4);"3. SELLING"
 1920 PRINTTAB(13,row+6);"4. SELL AND RE
BUY"
 1930 PRINTTAB(13,row+8);"5. NEW ISSUES"
 1940 PRINTTAB(13,row+10);"6. QUIT"
 1950 PRINTTAB(0,row+12)"Select a number
:"
 1960 REPEAT:N%=GET-48:UNTIL N%>0 AND N%
<7
 1970 IF N%=1 THEN PROCbuy
 1980 IF N%=2 THEN PROCinvest
 1990 IF N%=3 THEN PROCsell
 2000 IF N%=4 THEN REM see part 2
 2010 IF N%=5 THEN REM see part 2
 2020 IF N%=6 THEN VDU26:CLS:exit%=TRUE
 2030 ENDPROC
 2040 :
 2050 DEF PROCreport(msg$,x1,val,f)
 2060 IF f=0 @%=&09 ELSE @%=&20009+f*&10
0
 2070 PRINT msg$;:PRINTTAB(x1-2)"= "val
 2080 @%=&10
 2090 ENDPROC
 2100 :
 2110 DEF PROCreport1
 2120 PROCreport("Number bought",28,Nr1,
0)
 2130 PROCreport("Purchase price, p",28,
Sp1,1)
 2140 PROCreport("Savings A/C interest,
%",28,D%/100,2)
 2150 PROCreport("Actual value, `",28,Co
st,2)
 2160 PROCreport("Break even price, p",2
8,FNeven(0),1)
 2170 PROCreport("Year Break even price,
p",28,FNeven(12),1)
 2180 PRINT:PRINTTAB(18)"B/even";:PRINTT
AB(30)"Loss Limit"
 2190 FOR Month=1 TO 12
 2200 @%=&09:PRINT"Month ";SPC(-(Month<1
0))Month;
 2210 @%=&20109:PRINTTAB(15)FNeven(Month
);TAB(33)E%/100*FNeven(Month)
 2220 NEXT
 2230 @%=10:ENDPROC
 2240 :
 2250 DEF PROCreport3
 2260 PROCreport("Year of purchase",28,Y
r1,0)
 2270 PROCreport("Month purchased",28,M1
,0)
 2280 PROCreport("Price paid, p",28,Sp1,
1)
 2290 PROCreport("Number bought",28,Nr1,
0)
```

```
 2300 PROCreport("Year sold",28,Yr2,0)
 2310 PROCreport("Month sold",28,M2,0)
 2320 PROCreport("Selling price, p",28,S
p2,1)
 2330 PROCreport("Number sold",28,Nr2,0)
 2340 PROCreport("Savings A/C interest,
%",28,D%/100,2)
 2350 PRINT
 2360 PRINTCHR$131;SPC10;STRING$(20,"*")
 2370 PRINT
 2380 PROCreport("Months held",28,Mo,0)
 2390 PROCreport("Original investment, `
",28,Cost,2)
 2400 PROCreport("Sale value, `",28,FNsa
leval,2)
 2410 PROCreport("Break even value, `",2
8,FNeven3(Mo),2)
 2420 CaInc=FNcash:IF CaInc<0 C$=CHR$136
+CHR$129 ELSE C$=CHR$130
 2430 PROCreport("Cash increase/decrease
"+C$,28,CaInc,2)
 2440 Rt=FNreturn:IF Rt<D%/100 C$=CHR$13
6+CHR$129 ELSE C$=CHR$130
 2450 PROCreport("Average Annual Return"
+C$,28,Rt,2)
 2460 @%=10:ENDPROC
 2470 :
 2720 DEF FNeven(m)
 2730 LOCAL NewCost,NewCo
 2740 NewCost=Cost*(1+D%/10000)^(m/12)
 2750 NewCo=NewCost*B%/10000
 2760 IF NewCo>A% =100*(NewCost+1.15*New
Co)/Nr1 ELSE =100*(NewCost+1.15*A%/100)/
Nr1
 2770 :
 2780 DEF FNeven3(m)=Cost*(1+D%/10000)^(
m/12)+Co
 2790 :
 2800 DEF FNcost(N,S,f)
 2810 Co=FNcommission(N,S)
 2820 sd=0:IF f sd=(INT(2*C%/1000000*S*N
+0.5))/2
 2830 IF f=1 =S*N/100+1.15*Co+sd ELSE =S
*N/100-1.15*Co
 2840 :
 2850 DEF FNmonths:LOCAL T
 2860 T=(12-M1)+M2+(12*(Yr2-(Yr1+1)))
 2870 IF T<=0 THEN T=1 ELSE =T
 2880 :
 2890 DEF FNreturn:=(FNsaleval-Cost)*120
0/(Mo*Cost)
 2900 :
 2910 DEF FNcash:=FNsaleval-Cost
 2920 :
 2930 DEF FNnumber
 2940 LOCAL n
 2950 n=100*Ca/Sp1/(1+C%/10000+1.15*B%/1
0000)
 2960 Co=n*Sp1/100*B%/10000
 2970 IF Co<A%/100 Co=A%/100:n=100*(Ca-1
.15*Co)/(1+C%/10000)/Sp1
 2980 =n
 2990 :
 3000 DEF FNcommission(N,S)
 3010 LOCAL c
 3020 c=N*S/100*B%/10000
 3030 IF c<A%/100 =A%/100 ELSE =c
 3040 :
 3050 DEF FNnewprice=100*(FNsaleval-1.15
*FNcommission(Nr2,Sp2)-(INT(2*C%/1000000
*Sp2*Nr2))/2)/(Nr2*(1+H/100))
 3060 :
 3070 DEF FNsaleval=FNcost(Nr2,Sp2,0)
 3080 :
 3090 DEF PROCcont1:LOCAL key$
 3100 PRINTTAB(0,22)CHR$134;"'P' to prin
t : Escape to Menu";
 3110 REPEAT:key$=GET$
 3120 UNTIL key$="P" OR key$="p"
 3130 PROCprint
 3140 ENDPROC
 3150 :
 3160 DEF PROCcont2:LOCAL key$
 3170 PRINTTAB(0,22)CHR$134;"'C' to cont
inue : Escape to Menu";
 3180 REPEAT:key$=GET$
 3190 UNTIL key$="C" OR key$="c"
 3200 ENDPROC
 3210 :
 3220 DEF PROCerror:VDU26:CLS
 3230 IF ERR=17 THEN PROCtitle:GOTO 140
 3240 REPORT:PRINT" at line ";ERL
 3250 ENDPROC
 3260 :
 3270 DEF PROCprint
 3280 PRINTTAB(0,22) SPC39;
 3290 INPUTTAB(0,22)"Share: " name$
 3300 PRINTTAB(0,22) SPC39;
 3310 INPUTTAB(0,22)"Date: " date$
 3320 *FX3,10
 3330 VDU1,27,1,108,1,20:PRINT:VDU1,27,1
,14
 3340 PRINTTAB(6)name$;TAB(25)date$:PRIN
T:PRINT
 3350 IF FL=1 THEN PROCreport1
 3360 IF FL=2 THEN PROCreport1
 3370 IF FL=3 THEN PROCreport3
 3380 IF FL=4 THEN PROCreport4
 3390 IF FL=5 THEN PROCreport5
 3400 VDU12:*FX3,0
 3410 PROCcont1
 3420 ENDPROC
```

Counting Rabbits

A useful educational program to teach very young children to count.

As a change from some of our recent educational programs, we describe a program which is aimed at very young children.

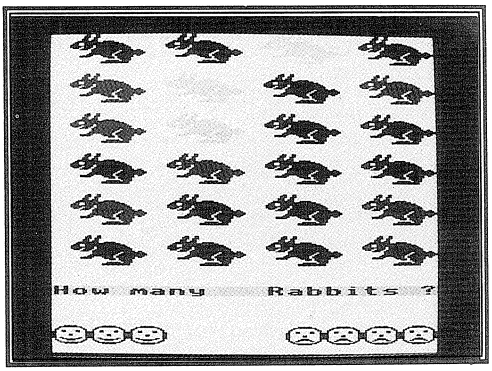
If you are looking for a means to introduce very young children to the computer, and at the same time make use of it as a teaching aid, then this program is just what you are looking for. To a young child the keyboard can be a very bewildering sight. The letter keys tend not to be of interest at first, which is not surprising when you consider how jumbled they are, but fortunately the number keys are in order and can easily be isolated as the top row.

Briefly this program is designed to teach counting of the numbers 0 to 9 to young children from the ages 3 to 6. 24 Rabbits in a random mix of 5 colours are displayed on a white background, which resembles the white paper children are used to drawing on. There are between 1 and 8 rabbits in each colour and the child is asked how many there are of a particular colour.

This basic question is repeated for all 5 colours. For each correct answer a happy face is printed at the bottom right, and for each wrong answer a sad face is printed at the bottom left. The correct number of the appropriate colour is also printed so the child can see where they went wrong.

After the 5 questions have been asked the colours of all the rabbits are changed and the process repeated until there is a continuous line of 10 faces at the bottom, whereupon the final score for correct and incorrect answers is printed. Pressing the space bar will repeat the process until Escape is pressed.

Recognising numbers is an important part of early education. It was for this purpose that the current version of this program was developed. An earlier version proved impossible for my own child to use without supervision as I was constantly having to read out the question asking "How many green (or red or blue etc.) Rabbits are there ?"



'Counting Rabbits' screen

By replacing the word for the colour by a background band, I was able to let my children work through the program unaided, which of course made them feel more important, and certainly stimulated their interest in completing the number of questions.

It is also important to introduce some sort of reward for successful achievement in any educational program. The introduction of the happy face (and sad face for an incorrect answer), along with an appropriate sound accompaniment gives added interest to the program.

PROGRAM NOTES

The program contains a number of REM lines which highlight the various segments. The

definitions for the Rabbit and Face shapes are placed in the user-defined ASCII characters 224 to 241 by PROCsetup.

The main program segment is from lines 280 to 570 with a FOR-NEXT loop between lines 300 and 530. The variables initialised in line 290 are as follows:

Xmin is the X position where the next happy face will be printed.

Xmax is the X position where the next sad face will be printed.

Sa is the number of sad faces, hence the number of wrong answers.

Sm is the number of smiling faces, hence the number of correct answers.

The array A() contains the numbers 1 to 5 which represent the colours red to magenta in mode 2. As each colour is selected, its number in the array is set to 0 to prevent it being selected again (line 480). This is necessary since the order of the colours in the questions is chosen at random to give some variety to the repeats of the program.

The array C() contains the correct number of rabbits of each colour and is used to check the answers.

In line 490 the number keys are read using the INKEY$ statement with a test for valid numbers (between 0 and 8).

PROCright is used when the correct number is pressed. This prints the answer to confirm the key pressed. A new happy face is printed to the right of any previous happy faces, and the happy tune is played.

PROCwrong is used when the wrong number key is pressed. This shows the correct answer. A new sad face is displayed to the left of any previous sad faces, and the sad tune is played.

```
   10 REM Program Counting Rabbits
   20 REM Version B1.1
   30 REM Author  Alan Pratt
   40 REM BEEBUG  May 1989
   50 REM Program subject to copyright
   60 :
  100 MODE2:ON ERROR GOTO600
  110 VDU23,1;0;0;0;0;
  120 VDU19,7,0,0,0,0:VDU19,0,7,0,0,0
  130 DIM A(5),C(5)
  140 FOR I=0 TO 5
  150 COLOURI-7*(I=0):PRINTTAB(2,I+2)"CO
UNTING RABBITS"
  160 NEXT
  170 COLOUR7
  180 PROCsetup
  190 FOR I=0 TO 16 STEP 4
  200 FOR J=0 TO 8 STEP 8
  210 PRINTTAB(I,J)CHR$224;CHR$225;CHR$2
24;CHR$225;TAB(I,J+1)CHR$226;CHR$227;CHR
$228;CHR$229
  220 NEXT:NEXT
  230 PRINTTAB(0,2)CHR$224;CHR$225;TAB(0
,3)CHR$228;CHR$229;TAB(0,4)CHR$224;CHR$2
25;TAB(0,5)CHR$226;CHR$227;TAB(0,6)CHR$2
24;CHR$225;TAB(0,7)CHR$228;CHR$229
  240 PRINTTAB(18,2)CHR$224;CHR$225;TAB(
18,3)CHR$226;CHR$227;TAB(18,4)CHR$224;CH
R$225;TAB(18,5)CHR$228;CHR$229;TAB(18,6)
CHR$224;CHR$225;TAB(18,7)CHR$226;CHR$227
  250 PROChappy_tune(-12,1.8):PROCsad_tu
ne(-10,3)
  260 COLOUR11:PRINTTAB(0,31)"Press the
Space Bar";:COLOUR7
  270 REPEAT UNTIL GET=32
  280 REPEAT
  290 CLS:Xmin=0:Xmax=18:Sa=0:Sm=0
  300 FOR D=1 TO 2
  310 FOR I=1 TO 5:A(I)=I:NEXT
  320 REM Reset number of each colour to
 0
  330 FOR I=1 TO 5:C(I)=0:NEXT:N=0
  340 FOR Y=0 TO 20 STEP 4
  350 FOR X=1 TO 16 STEP 5
  360 REM Choose colour of each Rabbit
  370 colour=RND(5)
  380 C(colour)=C(colour)+1:IF C(colour)
>9 THEN C(colour)=C(colour)-1:GOTO 370
  390 COLOUR A(colour):PRINTTAB(X,Y)CHR$
230;CHR$231;CHR$232;CHR$233
  400 PRINTTAB(X,Y+1)CHR$234;CHR$235;CHR
```

```
$236;CHR$237
  410 PRINTTAB(X,Y+2)CHR$238;CHR$239;CHR
$240;CHR$241
  420 NEXT:NEXT
  430 TIME=0:REPEAT UNTIL TIME>35
  440 REM Pick random colour
  450 REPEAT
  460 REPEAT:P=RND(5):UNTIL A(P)>0
  470 COLOUR7:COLOUR(A(P)+128):PRINTTAB(
0,25)"How many Rabbits ?  ":COLOUR128
  480 A(P)=0:*FX21,0
  490 REPEAT:guess=ASC(INKEY$(0))-48:UNT
IL guess>=0 AND guess<=8
  500 answer=C(P):IF guess=answer THEN P
ROCright ELSE PROCwrong
  510 TIME=0:REPEAT UNTIL TIME=100
  520 N=N+1:UNTIL N=5
  530 NEXT D
  540 IF Sm=10 THEN PRINTTAB(0,25)"You g
ot all 10 right  Congratulations" ELSE P
RINTTAB(0,25)"You got ";Sm;" right and
       ";Sa;" wrong    "
  550 COLOUR11:PRINTTAB(0,31)"Press the
Space Bar";:COLOUR7
  560 REPEAT UNTIL INKEY-99
  570 UNTIL FALSE
  580 END
  590 :
  600 MODE7:REPORT:PRINT" at line ";ERL
  610 END
  620 :
 1000 DEF PROCright
 1010 PRINTTAB(2,26)"That is correct."SP
C6"There ";:IF answer>1 THEN PRINT"are "
;answer ELSE PRINT"is ";answer
 1020 REM Print a smiling face
 1030 PRINTTAB(Xmin,29)CHR$224;CHR$225;T
AB(Xmin,30)CHR$226;CHR$227;
 1040 Xmin=Xmin+2:Sm=Sm+1
 1050 PROChappy_tune(-10,2)
 1060 TIME=0:REPEAT UNTIL TIME>360
 1070 PRINTTAB(0,26)SPC(39)
 1080 ENDPROC
 1090 :
 1100 DEF PROCwrong
 1110 PRINTTAB(2,26)"That is wrong."SPC8
"There ";:IF answer>1 THEN PRINT"are ";a
nswer ELSE PRINT"is ";answer
 1120 REM Print a sad face
 1130 PRINTTAB(Xmax,29)CHR$224;CHR$225;T
AB(Xmax,30)CHR$228;CHR$229;
 1140 Xmax=Xmax-2:Sa=Sa+1
 1150 PROCsad_tune(-9,3.5)
 1160 TIME=0:REPEAT UNTIL TIME>360
 1170 PRINTTAB(0,26)SPC(39)
 1180 ENDPROC
 1190 :
 1200 DEF PROChappy_tune(Amp,w)
 1210 SOUND 1,Amp,101,3*w:SOUND 1,Amp,11
7,w:SOUND 1,Amp,109,2*w:SOUND 1,Amp,121,
w:SOUND 1,Amp,117,3*w:SOUND 1,Amp,101,5*
w
 1220 ENDPROC
 1230 :
 1240 DEF PROCsad_tune(Amp,w)
 1250 SOUND 1,Amp,81,1.5*w:SOUND 1,Amp,6
9,w:SOUND 1,Amp,53,1.5*w:SOUND 1,Amp,69,
w:SOUND 1,Amp,61,.8*w:SOUND 1,Amp,61,2*w
 1260 ENDPROC
 1270 :
 1280 DEF PROCsetup
 1290 REM Set up the graphic characters
 1300 VDU23,224,7,31,48,96,76,204,192,19
3
 1310 VDU23,225,224,248,12,6,50,51,3,131
 1320 VDU23,226,193,192,216,79,99,48,31,
7
 1330 VDU23,227,131,3,27,242,198,12,248,
224
 1340 VDU23,228,193,192,195,71,108,48,31
,7
 1350 VDU23,229,131,3,195,226,54,12,248,
224
 1360 VDU23,230,1,3,3,7,7,31,63,115
 1370 VDU23,231,152,152,184,48,112,243,2
55,255
 1380 VDU23,232,0,0,0,0,0,224,248,254
 1390 VDU23,233,0,0,0,0,0,0,0,0
 1400 VDU23,234,243,255,251,119,15,0,0,0
 1410 VDU23,235,255,223,223,191,127,127,
127,255
 1420 VDU23,236,255,255,255,255,255,251,
247,239
 1430 VDU23,237,0,128,192,192,236,254,25
5,255
 1440 VDU23,238,1,15,15,0,0,0,0,0
 1450 VDU23,239,255,199,131,0,0,1,7,7
 1460 VDU23,240,223,223,239,239,7,251,25
5,255
 1470 VDU23,241,254,236,224,192,192,192,
128,128
 1480 ENDPROC
```

Termin Review

Peter Rochford looks at a new viewdata terminal emulator.

Product	Termin
Supplier	Cyansoft Videotex, 224a Chatham Hill, Chatham, Kent ME5 7BA. Tel. (0634) 570188
Price	£19.95 inc. VAT

There are so many comms software packages already available for the BBC micro, it is surprising to discover that yet another one has been released. However, one has - in the shape of *Termin* from a company called Cyansoft Videotex.

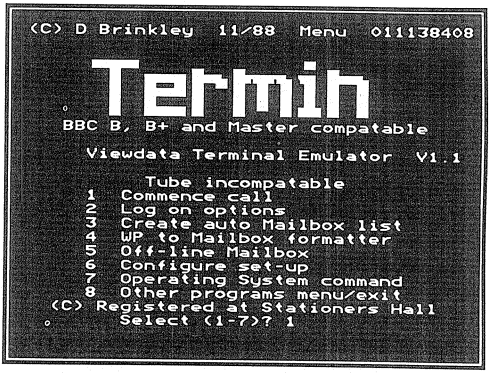
Termin is available for all the BBC computer range (excluding the Archimedes), and is supplied on either a single 80 track disc or two 40 track discs. The documentation takes the form of a 105 page wire bound manual, and there are also three rather poorly produced cut-out keystrips. Although described as a viewdata terminal emulation package, *Termin* does also feature a scrolling ASCII text terminal, too.

Using *Termin* is relatively straightforward. The software is largely menu-driven when off-line, and selections may be made by number, cursor keys, or a mouse. An unusual feature is the use of speech for various prompts when on-line. Unfortunately, this only works with the original Acorn speech upgrade, and this can only be fitted to the model B and B+. It is a pity that the program cannot be configured to work with, say, the Superior Software Speech system.

Both manual and auto-dial modems are catered for by *Termin*, and there are three different auto-dialling modes supported. These are suitable for the Pace Nightingale, Hayes compatible modems (for example the Pace Linnet), and any modem that auto-dials by pulsing the Request to Send line on the serial port. This latter category includes the Demon, the Voyager and the many budget modems with an auto-dial facility. There doesn't appear to be anyway of adding other modem drivers, and therefore if your modem does not appear in the above list you will have to resort to manual dialling.

Termin supports all standard baud rates up to 2400/2400 (V22 bis).

Before using *Termin* you must configure it to suit your own particular set-up and needs. Once done, these configuration options are saved to disc and become the default each time the package is used, unless manually overridden.



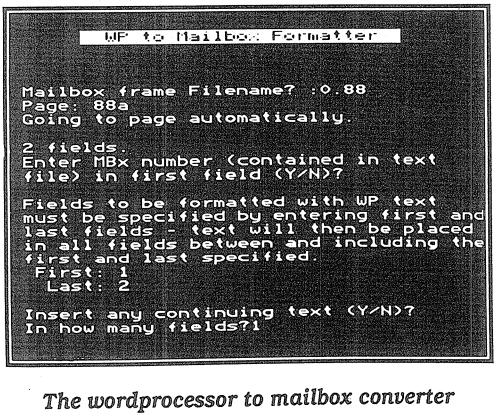
Termin's main menu

The software will allow easy log-on to any database by single key press. You set up a list of your favourite databases in a directory, along with their telephone numbers and your ID and password to enter the system. When an entry is selected, the software will then auto-dial it and log you on automatically, even re-sending your ID and password automatically if requested because of corruption by line noise. Naturally, this does require an auto-dial modem to use this feature.

Like most other viewdata software around, *Termin* features telesoftware downloading and sends downloaded files straight to disc. An unusual facility offered by *Termin* however, is the ability to download several files automatically and then log-off. This sounds good in principle, but in practice involves knowing page numbers, routes and length of header frames. By the time you have sorted all that out you might just as well have done it all manually!

Another automatic facility in *Termin* is mailbox download. Once logged on to, say, Prestel, pressing f6 will cause the software to collect all your waiting mailboxes and send them to disc, after which it will log you off. The file of frames created on disc can then be viewed when offline or printed. A further option allows the complete file of frames to be stripped of colour and graphics codes, ready for loading into a word processor.

Still on the subject of mailboxing, yet another automatic option in *Termin* is 'mailbox mailshot'. This feature enables the user to send the same mailbox to any number of people on Prestel. The mailbox number list must first be created in a word processor or could be taken from the directory on Prestel. Then you create your mailbox frame, go online, and the software when requested will do the rest. Great for junk mailing, though unlike some auto-mailers, *Termin* will not automatically remove numbers which result in no response.



The wordprocessor to mailbox converter

Termin enables both offline and online mailbox editing. Offline editing can utilise any mailbox frame, which once created, can then be uploaded onto Prestel with the frame-send function.

In common with even the most basic viewdata software, *Termin* allows the saving, loading and printing of any viewdata frame. The saved frames are stored in files which can hold up to 10 frames. When using a Master, the files are automatically time and date stamped to aid in future reference. *Termin* also features a call charge recorder, a facility normally only found on dearer packages.

I have already mentioned that the software has a mailbox editor. In addition, it also has a far more sophisticated viewdata frame editor. Why it needs two frame editors I don't know! Surely one would suffice to allow editing at any level of complexity by the user, depending on their needs. Nevertheless, a frame editor is provided and boasts quite a large number of features. You may use it to cut, paste and copy blocks of text and graphics within a frame, or save and load blocks. For text processing, you can set up a window within a frame, and text within that window will be automatically formatted when entered. Word processor files can be imported into the frame and again they will be automatically formatted.

As mentioned at the start of this review, *Termin* is not just a viewdata terminal emulator. It also features a scrolling text terminal that is quite respectable. File transfer facilities are provided and it supports Xmodem as well as ASCII upload/download either to RAM buffer or direct to disc. As with the viewdata terminal, the text terminal can be configured to suit the user and enables the same single keypress auto-dial and logon facility.

DOCUMENTATION

The manual I would describe as reasonably good, but it could certainly do with more examples to illustrate the use of the various terminal features. Raw beginners to comms would definitely struggle in quite a few instances. It is written in an informal chatty style, and has the occasional somewhat witty comment thrown in. More amusing however, are the numerous misprints, for example you are quite often asked to press the 'APE' key!

CONCLUSION

Any heavy criticism of this package would be churlish when you consider its price. It has a large number of features, some of which are not found on more expensive comms software. Certain features are questionable as to their real value, but the package includes all of the basics required for a competent viewdata terminal with a scrolling text terminal thrown in, too.

I found *Termin* pleasant to use and would recommend it. B

Speed and Sound Controller

This utility from Ian Kelly gives you more control over the execution of your programs.

Have you got your highest score in a game only to hear the doorbell ring? Are your games too fast and noisy? Or do you have to strain to see the last bit of text printed before the entire program crashes?

The utility given here will allow you to slow your computer down, pause programs, and turn the sound on and off. It comes in the shape of a ROM image, which can either be loaded into sideways RAM, or blown into an EPROM and installed permanently.

ENTERING THE PROGRAM

The listing should be entered and saved before running it. When run, the ROM image is assembled and saved under the name 'SLOWROM'. Before use, the image must be loaded into sideways RAM. This can be done with the following command:

```
*SRLOAD SLOWROM 8000 ZQ
```

Once loaded, you should press Ctrl-Break to initialise the ROM image.

USING THE SLOWROM

Once installed, the SLOWROM offers five new commands. These are:

```
*SLOW <num>
*QUIET
*SOUND
*CHEAT
*NOCHEAT
```

The command *HELP SLOW will list these commands together with the current status of the ROM. This will be self explanatory once the functions of the commands are understood.

***SLOW <num>**

This command changes the speed at which programs are executed. The number should be in the range 0 to 9. A value of zero indicates normal speed, while a setting of 9 slows the computer down to about a quarter of normal speed. The other values have effects between these two extremes.

***QUIET**

This turns off all sound.

***SOUND**

This reverses the effect of *QUIET, turning the sound back on.

***CHEAT**

This commands enables any program to be subsequently frozen by pressing the Caps Lock and Shift Lock keys together. The name arises from the usefulness of this feature to freeze a game and give you thinking time. Additionally, when the program is frozen it is possible to change the speed of execution and turn the sound on and off.

Once frozen, the speed can be changed by pressing any key from '0' to '9'. Alternatively, the sound can be turned on and off using 'S' and 'Q' respectively. Execution is resumed by pressing (and releasing) the space bar.

***NOCHEAT**

This turns off the cheat mode.

The SLOWROM uses a location of the CMOS RAM (or EEPROM) to store the current setting of the speed, and the cheat and sound enables. This ensures that if the computer is turned off, whenever the SLOWROM is reloaded the previous settings are restored.

HOW IT WORKS

The operation of the SLOWROM is relatively straightforward, with most of the program involved with decoding the star commands. The key to the program is a ROM service call, which when enabled, is issued 100 times a second. The SLOWROM intercepts this call and uses it for two purposes. Firstly, if the execution speed is set to non-zero, the SLOWROM sits in a loop decrementing a counter. The initial value of the counter depends on the current speed setting, and this serves to 'waste' a variable amount of time, hence slowing down execution.

The second use of the service call is to test if cheat mode is enabled, and if it is, whether the two Lock keys are pressed. If this is the case, the SLOWROM then scans the keyboard respectively waiting for a key press. If '0' - '9', 'S' or 'Q' are pressed, then the appropriate action is taken. On the other hand, if the space bar is pressed then the routine exits, returning control to the program.

The SLOWROM uses memory locations &AE, &AF as workspace, and CMOS RAM (EEPROM) location 30. This should cause no compatibility problems, and the SLOWROM will even work with most commercial games.

```
 10 REM Program Slow ROM
 20 REM Version 2.20
 30 REM Author  Ian Kelly
 40 REM BEEBUG  May 1989
 50 REM Program subject to copyright
 60 :
100 osbyte=&FFF4:osasci=&FFE3
110 DIM code &400
120 FORpass=4 TO 7 STEP 3
130 P%=&8000:O%=code
140 [OPT pass
150 EQUB 0:EQUB 0:EQUB 0
160 JMP check
170 EQUB &82
180 EQUB copyright MOD256
190 EQUB 1
200 EQUS "Slow":EQUB 0
210 EQUS "2.20"
220 .copyright
230 EQUB 0
240 EQUS "(C) BEEBUG 1989":EQUB 0
250 :
260 .check
270 PHP:PHA:PHX:PHY
280 CMP #4:BNE check1
290 JMP checkcommand
300 .check1
310 CMP #9:BNE check2
320 JMP checkhelp
330 .check2
340 CMP #21:BNE check3
350 JMP slowmicro
360 .check3
370 CMP #39:BNE end
380 JMP break
390 .end
400 PLY:PLX:PLA:PLP
410 RTS
420 :
430 .helpmess
440 EQUB 13
450 EQUS "Slow 2.20":EQUB 13
460 EQUB 32:EQUB 32
470 .commandtable
480 EQUS "SLOW":EQUB 0
490 EQUB slow MOD256
500 EQUB slow DIV256
510 EQUS "CHEAT":EQUB 0
520 EQUB cheat MOD256
530 EQUB cheat DIV256
540 EQUS "NOCHEAT":EQUB 0
550 EQUB nocheat MOD256
560 EQUB nocheat DIV256
570 EQUS "QUIET":EQUB 0
580 EQUB quiet MOD256
590 EQUB quiet DIV256
600 EQUS "SOUND":EQUB 0
610 EQUB sound MOD256
620 EQUB sound DIV256
630 EQUB &FF
640 :
650 .checkcommand
660 STY &AF
670 LDX #0:STX &AE
680 .checkcommandlp
690 LDA commandtable,X
700 BNE nozero
710 LDA &AE:BEQ foundcomm
720 INX:INX:INX
730 LDY &AF
740 LDA #0:STA &AE
750 LDA commandtable,X
760 CMP #&FF:BNE checkcommandlp
770 JMP end
780 .nozero
790 CMP (&F2),Y
800 BEQ equal
810 .notcomm
820 LDA #1:STA &AE
830 .equal
840 INX:INY
850 JMP checkcommandlp
860 .foundcomm
870 LDA (&F2),Y
880 CMP #13:BEQ foundcomm1
890 CMP #32:BEQ foundcomm1
900 JMP notcomm
910 .foundcomm1
920 LDA commandtable+1,X:STA &AE
930 LDA commandtable+2,X:STA &AF
940 JMP (&AE)
950 :
960 .slow
970 LDA (&F2),Y:CMP #32:BNE perzer
```

```
 980 .slowspclp
 990 INY
1000 LDA (&F2),Y:CMP #32:BEQ slowspclp
1010 CMP #48:BCC perzer
1020 CMP #58:BCS perzer
1030 .changeslow
1040 JSR changespeed
1050 .clend
1060 PLY:PLX:PLA:PLP
1070 LDA #0:RTS
1080 .perzer
1090 LDA #48:JMP changeslow
1100 .changespeed
1110 SEC:SBC #48:STA &AF:BEQ noslow
1120 LDA #161:LDX #30:JSR osbyte
1130 TYA:TAX:AND #64:BNE pollingalready
1140 TXA:CLC:ADC #64:TAX
1150 LDA #22:JSR osbyte
1160 .pollingalready
1170 TXA:AND #240:CLC:ADC &AF
1180 TAY:LDA #162:LDX #30:JSR osbyte
1190 RTS
1200 .noslow
1210 LDA #161:LDX #30:JSR osbyte
1220 TYA:TAX:AND #64:BEQ endcs
1230 LDA #23:JSR osbyte
1240 TXA:AND #160
1250 TAY:LDA #162:LDX #30:JSR osbyte
1260 .endcs
1270 RTS
1280 :
1290 .cheat
1300 LDA #161:LDX #30:JSR osbyte
1310 TYA:TAX:AND #127:BNE cheat1
1320 LDA #22:JSR osbyte
1330 .cheat1
1340 TXA:AND #127:CLC:ADC #128
1350 TAY:LDA #162:LDX #30:JSR osbyte
1360 JMP clend
1370 :
1380 .nocheat
1390 LDA #161:LDX #30:JSR osbyte
1400 TYA:TAX:AND #127:BNE nocheat1
1410 LDA #23:JSR osbyte
1420 .nocheat1
1430 TXA:AND #127
1440 TAY:LDA #162:LDX #30:JSR osbyte
1450 JMP clend
1460 :
1470 .quiet
1480 JSR nosound
1490 JMP clend
1500 :
1510 .nosound
1520 LDA #161:LDX #30:JSR osbyte
1530 TYA:AND #223:CLC:ADC #32
1540 TAY:LDA #162:LDX #30:JSR osbyte
1550 LDA #210:LDX #1:LDY #0:JSR osbyte
1560 RTS
1570 :
1580 .sound
1590 JSR soundon
1600 JMP clend
1610 :
1620 .soundon
1630 LDA #161:LDX #30:JSR osbyte
1640 TYA:AND #223
1650 TAY:LDA #162:LDX #30:JSR osbyte
1660 LDA #210:LDX #0:LDY #0:JSR osbyte
1670 RTS
1680 :
1690 .checkhelp
1700 LDA (&F2),Y
1710 CMP #13:BNE help
1720 LDX #0
1730 .genhelplp
1740 LDA helpmess,X:JSR osasci
1750 INX
1760 CMP #0:BNE genhelplp
1770 LDA #13:JSR osasci
1780 .endhelp
1790 JMP end
1800 :
1810 .help
1820 CMP #32:BNE nospaces
1830 .ridspaceslp
1840 INY
1850 LDA (&F2),Y:CMP #32
1860 BEQ ridspaceslp
1870 .nospaces
1880 LDX #0
1890 .helplp
1900 LDA (&F2),Y:CMP #13:BNE notend
1910 LDA #0
1920 .notend
1930 CMP commandtable,X:BNE endhelp
1940 INX:INY
1950 CMP #0:BNE helplp
1960 LDX #0
1970 .helpprintlp
1980 LDA helpmess,X
1990 CMP #&FF:BEQ nomorecomms
2000 JSR osasci:INX
2010 CMP #0:BNE helpprintlp
2020 INX:INX
2030 LDA #13:JSR osasci
2040 LDA #32:JSR osasci:JSR osasci
2050 BRA helpprintlp
2060 .nomorecomms
2070 CMP #&FF:BNE helpprintlp
2080 LDX #0
2090 .statuslp
```

```
2100 LDA statusmess,X:JSR osasci
2110 INX:CPX #42:BCC statuslp
2120 LDA #ASC("O"):JSR osasci
2130 LDA #161:LDX #30:JSR osbyte
2140 TYA:AND #128:BNE con
2150 LDA #ASC("f"):JSR osasci
2160 JSR osasci
2170 LDA #8:JSR osasci
2180 JMP slsp
2190 .con
2200 LDA #ASC("n"):JSR osasci
2210 .slsp
2220 LDA #8:JSR osasci:JSR osasci
2230 LDA #10:JSR osasci
2240 TYA:AND #15:CLC:ADC #48:JSR osasci
2250 LDA #8:JSR osasci
2260 LDA #10:JSR osasci
2270 LDA #ASC("O"):JSR osasci
2280 TYA:AND #32:BEQ son
2290 LDA #ASC("f"):JSR osasci
2300 JSR osasci
2310 JMP soff
2320 .son
2330 LDA #ASC("n"):JSR osasci
2340 .soff
2350 LDA #13:JSR osasci
2360 JMP end
2370 .statusmess
2380 EQUB 13:EQUS "Cheat Mode :"
2390 EQUB 13:EQUS "Speed      :"
2400 EQUB 13:EQUS "Sound      : "
2410 EQUB 11:EQUB 11
2420 :
2430 .slowmicro
2440 LDA #161:LDX #30:JSR osbyte
2450 TYA:AND #64:BEQ notslow
2460 TYA:AND #15:TAY
2470 LDA speedtable,Y:TAY
2480 .slowmicrolp
2490 LDX #6
2500 .slowmicrolp1
2510 NOP
2520 DEX:BNE slowmicrolp1
2530 DEY:BNE slowmicrolp
2540 :
2550 .notslow
2560 LDA #161:LDX #30:JSR osbyte
2570 TYA:AND #128:BNE keys
2580 JMP end
2590 :
2600 .keys
2610 LDA #&81:LDY #&FF:LDX #175
2620 JSR osbyte:TXA:BEQ nothold
2630 LDA #&81:LDY #&FF:LDX #191
2640 JSR osbyte:TXA:BEQ nothold
2650 .holdlp
2660 LDY #0
2670 .tsplp
2680 PHY
2690 LDA keytable,Y
2700 TAX:LDA #&81:LDY #&FF
2710 JSR osbyte:PLY:TXA:BEQ notkey
2720 TYA:CLC:ADC #48
2730 JSR changespeed
2740 JMP holdlp
2750 .notkey
2760 INY
2770 CPY #10:BCC tsplp
2780 LDA #&81:LDY #&FF:LDX #174
2790 JSR osbyte:TXA:BEQ testquiet
2800 JSR soundon
2810 JMP holdlp
2820 .testquiet
2830 LDA #&81:LDY #&FF:LDX #239
2840 JSR osbyte:TXA:BEQ testend
2850 JSR nosound
2860 JMP holdlp
2870 .testend
2880 LDA #&81:LDY #&FF:LDX #157
2890 JSR osbyte:TXA:BEQ holdlp
2900 .testend2 LDA #&81:LDY #&FF
2910 LDX #157:JSR osbyte
2920 TXA:BNE testend2
2930 .nothold
2940 JMP end
2950 .speedtable
2960 EQUB 0:EQUB 28:EQUB 56:EQUB 84
2970 EQUB 112:EQUB 140:EQUB 168
2980 EQUB 196:EQUB 224:EQUB 255
2990 .keytable
3000 EQUB 216:EQUB 207:EQUB 206
3010 EQUB 238:EQUB 237:EQUB 236
3020 EQUB 203:EQUB 219:EQUB 234
3030 EQUB 217
3040 :
3050 .break
3060 LDA #161:LDX #30:JSR osbyte
3070 TYA:TAX:AND #128:BEQ break1
3080 LDA #22:JSR osbyte
3090 .break1
3100 TXA:AND #64:BEQ break2
3110 LDA #22:JSR osbyte
3120 .break2
3130 TXA:AND #32:BEQ break3
3140 LDA #210:LDX #1:JSR osbyte
3150 .break3
3160 JMP end
3170 ]
3180 NEXT
3190 OSCLI ("SAVE SLOWROM "+STR$~code+
" "+STR$~O%)
```

BEEBUG Education

by Mark Sealey

Cynics sometimes say that the real money in educational software is in programs which make life easier for teachers. This month BEEBUG Education looks at two such suites: a statistics package that is as flexible as most currently available, and software to develop and write pupil reports and assessments. We shall be becoming increasingly used to this latter operation in years to come!

Product	**FIRST**
Supplier	**Serious Statistical Software Lynwood, Benty Heath Lane, Willaston, South Wirral L64 1SD.**
Price	**£150 (no VAT)**

To scan the publicity material for this excellent product is like reading the index of a textbook on statistics. The strongest point of the software (available for the BBC 'B', B+, Master and the Archimedes) is its comprehensiveness. Although not designed to hold multi-way tables applicable to large surveys, for instance, most other statistical functions are possible with FIRST.

This includes a variety of regression facilities (weighted, standardised, residuals, confidence intervals and resultant contour plots), regression diagnostics: variance inflation factors, influential points and Cooks distances, robust regression, non-linear least squares as well as time series techniques. This list is not by any means exhaustive.

It ranges from common coefficients to highly specialist routines impossible to name in a short review. Suffice to say that functions tested during the review period performed well and should satisfy the needs of most situations. Anyone compiling data on educational performance, for instance, and wanting to identify which factors may influence it, should look closely at this product.

FIRST is in use in several University Departments as well as Government Offices and even commercial organisations. Although not originally intended for the educational market, a number of features make it particularly applicable for teaching at the higher levels (FE and HE and as a research tool).

The first of its features to recommend is its modular structure. Although written in a low-level subset of Basic for portability's sake (there are versions for PCs too now), it has been easy to add routines to perform specific tasks, as this has been requested of its enthusiastic developer, who is also keen to help individual customers... FIRST is in a continual state of update.

Not that the suite is in any way bitty. The logic of data entry, as well as controlling processing and presentation, is consistent and relatively easy to get to grips with. Though it has to be said at this point that FIRST is nowhere near as user-friendly as perhaps it should be. Response to prompts (which are nevertheless usually unambiguous and simple) is only accepted in upper case! The routine for loading files lacks the forgiving nature of a more sophisticated front-end.

This could well be very off-putting for the novice or inexperienced computer-user. It is doubtful whether the quite legitimate claim made by the publisher, that it is aimed at the "serious professional level", really lessens the impact of these shortcomings. To be fair, such occurrences as division by zero, are well trapped and you are always returned from other sources of potential disaster to the 'control-mode' prompt.

This prompt allows you to issue up to fifty two letter commands to control both functions and plotting of data once entered, as well as the generation of data itself. For example, once the data has been read in or entered, "LRW" performs Linear Regression Weighted by any variable. There are sample files on disc, and the

plotting (of any pair of variables in whatever set you are using) can be in mode 0 as well as 7.

The manual is clear and thorough, and contains indexing not only to pages but also to menu functions within the program. There is a list of useful references to statistical texts and many examples.

In summary, this is a clear best buy from amongst statistical tools of its type and can also be safely recommended for the teaching of statistics.

Product	**Assessment and Report Writing**
Supplier	**Ashford Press publishing 1 Church Road, Shedfield, Hants SO3 2HW.**
Price	**£46 incl VAT**

Just when you thought there would be no more mention of the National Curriculum in these columns, along comes a review of a good package designed to make administration easier by streamlining the chore of writing reports on pupils. This software is sufficiently flexible, that these reports can take the form of short formal assessments, to reveal strengths and weaknesses to both pupil and teacher. Or they can produce more prose-based material of a type usually given to pupils' families.

Although the present review was written with only the benefit of the demonstration disc (which is available for £4.60 inc. VAT), it is clear that the way it links pupils to subject-sets or to classes, for example, will make the job of compiling reports less painful than having stacks of paper scattered around - as is often the case at this time of year!

The software works on a system of easy-to-use menus and submenus, with single letter choices for most things. Text can also be output to a word processor, and the publishers claim that finishing one report can take as little as two minutes. The other side of this coin is that assessment statements, as they appear, are actually selected from a number of options and inserted into the report: "Has not fully understood the material", for example. Admittedly, the user can select which criteria to work with (level of understanding, attitude to the subject, application and the rest). Alternatively, sample assessment criteria to match those dictated by the National Curriculum are in preparation. They will thus be updated as these change. What is more, several copies of the disc for different departments can be made without infringing copyright.

If you tend to say broadly the same sort of things about your pupils, then none of this will worry you. However, where you prefer to sit down and draft what might in some circumstances be quite an influential document, according to impressions and recollections, then this software will be far too restricting.

On the other hand, reports are compiled according to marks and grades in a way over which you have complete control. If you feel that a finite number of statements from which to chose (typically a dozen or so) is sufficient, and about half as many gradations in how the assessment criteria are met, then sit back and let the software do the donkey work. With one or two minor reservations (which were probably associated with the fact that the full version was not reviewed), it is easy enough to use.

The verdict? It seems very likely that more programs of this kind are set to appear if the framework of the National Curriculum does take hold. As more and more pressure is put on teachers to spend time assessing pupils, rather than enabling their learning, many will find this sort of prop very welcome. If you are attracted to the idea of quite mechanical extrapolation of pupil records and collation in a variety of forms, then this suite is a serviceable example of what is needed.

Importantly, its relative flexibility means that comments can also be made which indicate strengths and weaknesses as well as performance. This is very much in the spirit of the infamous TGAT Report, which stressed the importance of assessments which, whilst not exactly diagnostic, were less judgemental than indicative of the pupils' future needs. Worth looking at. B

Big Text Screen Displays

Robin Murphy explains how programs can be made to display text automatically in double-height characters in any graphics mode.

This program was developed for use with the computer in the classroom, where the normal height text displays are not always as readable as one might wish, however the program may equally be applied in any circumstances where larger size text is desirable. Mode 7 does allow double-height text to be displayed, though not in immediate mode, but even then programming changes will be needed to achieve the larger size.

The accompanying program (Big Text) is designed to provide a facility which will work in any mode, except mode 7 (where characters cannot be redefined), and will convert normal height to double height text on the screen. It can be used either to affect text entered in immediate mode, or as a forerunner to other programs which will then function with Big Text. The resulting screen displays are certainly much easier to read.

Type in the program and save a copy to disc or tape. Then run the program and save the resulting machine code by typing:

```
*SAVE BIG A00 B00
```

The Big Text routine can be re-installed at any time by typing:

```
*BIG      (or *RUN BIG from tape)
```

In fact, you will find that Big Text has been enabled as soon as assembly is complete. It will be temporarily disabled whenever mode 7 is selected, and permanently disabled by pressing Break or Ctrl-Break.

A further way to disable Big Text is the procedure PROCunbig included at the end of the program listing. This is not used by the program as such, but may be included in any other program which needs to switch Big Text on and/or off. Once deselected, Big Text can be reinstated by using:

```
CALL &A00
```

The program as listed also replaces the character '0' by the letter 'O', as I find this better. If you do not wish to implement this feature then simply omit lines 1440 to 1490. On the other hand, similar code can easily be inserted to convert any other characters as you wish.

A further comment is necessary if text is to be displayed (in a program) at the graphics cursor after using VDU5. If the cursor is at the start of a line when VDU5 is executed, an extra line feed may be output causing the Big Text to slip down the screen. This can be avoided by replacing the relevant VDU5 by VDU9,5 which moves the cursor one space right before entering graphics mode. Otherwise, any VDU codes (codes 0 to 31) are catered for in the Big Text routine so that they function correctly.

PROGRAM NOTES

Essentially, the program forms a patch to a redirected OSWRCH routine. The main features of the program are as follows:

Lines Explanation

1020 - Assembly at page &A, but this could be any free page of RAM.
1040 - Redirect the WRCH vector to BIG
1090 - List of VDU codes which must be followed by extra bytes.
1130 - The number of extra bytes or zero for UP (11) or DOWN (12).
1170 - Start of BIG routine proper, accumulator holds character to print.
1180 - Flag>0 ignore byte and print normally.
1190 - Flag=0 preserve X & Y registers.
1220 - Read mode and ignore 7 or 135.
1280 - VDU code 0 to 31 - check list.
1310 - Double height delete.
1440 - Conversion of '0' to 'O'.
1490 - Use OSWORD with A=135 to read definition of current character into next 8 bytes of data_buffer.
1540 - Define CHR$255 to be top/bottom halves of expanded character using VDU23, and print.

1750 - Prepare to print additional DOWN if at start of line.
1780 - Normal ending - print current character and .....
1800 - restore X & Y registers.
1830 - Print using normal WRCH vector
1850 - comparison with list of special VDU codes - no match means print as normal. Match with UP or DOWN means print twice, otherwise print required number of bytes.

```
  10 REM Program Big Text
  20 REM Version B1.1
  30 REM Author  Robin Murphy
  40 REM BEEBUG  May 1989
  50 REM Program subject to copyright
  60 :
 100 PROCassemble
 110 CALL initialise
 120 MODE2
 130 PRINTTAB(0,10)"Assembly complete."
 140 END
 150 :
1000 DEFPROCassemble
1010 FOR opt=0 TO 3 STEP 2
1020 P%=&A00
1030 [OPT opt
1040 .initialise \ set WRCHVector
1050 LDA# diversion MOD 256:STA &20E
1060 LDA# diversion DIV 256:STA &20F
1070 RTS
1080 .flag  EQUB 0
1090 .codes EQUD &1F161101
1100        EQUD &131D1C12
1110        EQUD &0A171819
1120        EQUB &0B
1130 .extra EQUD &02010101
1140        EQUD &05040402
1150        EQUD &00090805
1160        EQUB &00
1170 .diversion \ OSWRCH patch
1180 DEC flag:BMI inc:JMP print
1190 .inc \ reset flag to 0
1200 INC flag:STA data_buffer
1210 TXA:PHA:TYA:PHA
1220 LDA#135
1230 JSR &FFF4  \ read MODE into Y
1240 TYA:AND #127:TAY
1250 LDA data_buffer
1260 CPY#7:BNE check_vdu
1270 JMP ending
1280 .check_vdu
1290 CMP #32:BCS delete
1300 JMP vdu_codes
1310 .delete
1320 CMP #127:BNE swap
1330 LDX &318 \ check for POS=0
1340 BNE cont
1350 LDA #11   \ move UP
1360 JSR print
1370 LDA #127
1380 .cont
1390 JSR print
1400 LDA #9:JSR print
1410 LDA #11:JSR print
1420 LDA #127:JSR print
1430 LDA #10:JMP ending
1440 .swap \ replace 0 by O
1450 CMP # ASC"0"
1460 BNE double_height
1470 LDA # ASC"O"
1480 STA data_buffer
1490 .double_height
1500 LDA #10
1510 LDY # data_buffer DIV 256
1520 LDX # data_buffer MOD 256
1530 JSR &FFF1 \ OSWORD read char def
1540 LDY #0
1550 .def_char
1560 LDA #23:JSR print
1570 LDA #255:JSR print
1580 .half_char
1590 INY
1600 LDA data_buffer,Y
1610 JSR print
1620 JSR print
1630 CPY #4:BEQ top_half
1640 CPY #8:BEQ bottom_half
1650 BNE half_char
1660 .top_half
1670 LDA #11:JSR print
1680 LDA #255:JSR print
1690 LDA #8     \ move DOWN
1700 JSR print
1710 LDA #10:JSR print
1720 JMP def_char
1730 .bottom_half
1740 LDA #255:JSR print
1750 LDX &318  \ check POS=0
1760 BNE restore
1770 LDA#10
```

Continued on page 28

Another Date with View

Jeff Gorman's short utility shows how the current date can be inserted automatically and permanently into a View text file.

When used on the Master, View allows the current date, obtained from the real-time clock, to be incorporated into any text as it is printed. This is achieved by including '|D' with any of the stored commands DH, DF, RJ, LJ or CE. The printout will show the date in the form:

```
nn mmm yyyy
```

for example:

```
05 Feb 1989
```

Similarly, "|T" generates the time as:

```
hh:mm:ss
```

such as:

```
19:10:05
```

With View in command mode, this can be verified by means of the SCREEN command, and both date and time are derived from constantly updated values in CMOS RAM. However, there is no easy way of inserting the date itself into a text file as a permanent record, yet this would often be useful. The program listed at the end of this article is designed to work with View to provide this facility.

The same information on date and time can be obtained in Basic from the pseudo variable:

```
TIME$
```

which returns the information in the form:

```
Day, nn mmm yyyy.hh:mm:ss
```

While the "nn mmm yyyy" format given by "|D" is adequate for some purposes, it does not match the more conventional format for general correspondence, which is:

```
5th February, 1989
```

so we will use this format in what follows.

The program DateProg modifies the date held in TIME$ and stores the result in memory so that Ctrl-Shift-f6 (unused by View) will insert the current date into the text in the format illustrated above.

Key f6 was chosen because with certain keys (f3, f7 or f9), forgetting to press Ctrl/Shift correctly could lead to unintentional deletion of text. However, you can easily change the key used by modifying line 130.

UNDERSTANDING THE PROGRAM

The main purpose of the program is to call a function FNdate which creates the date as a string. The function begins by using MID$ to skip the day element of TIME$, and extract the two numerals of the date into the string variable called *date$*. Converting this to a number (variable date) and back to string form gets rid of any leading zero, and provides a value for the following IF statements.

While the numerals(s) of the date are most commonly followed by the suffix "th", this is not invariably the case. String variable *suf1$* is first primed with "th", but the next three lines examine the date to decide whether "th" should be replaced by "st", "nd" or "rd".

The variable *mth$* is then assigned the "mmm" part of TIME$, which is then compared with the first three characters of the months in the data list at the end of the routine. Once matched, the full month name is substituted. If you do not wish to include the comma after the month, then omit this from the end of line 1150.

Finally the variable *year$* is set using MID$(TIME$,12,4), and by adding *date$* to *mth$* and *year$* the fully-fledged form of the date is established as the string returned by FNdate.

USING THE FUNCTION

The difficulty in using Basic programs to enhance View, is that the two programs cannot co-exist. However, the memory devoted to the function key definitions is not affected by the switch from Basic to View, so one can create the date string as a function key definition.

This is done in the main part of the program where *FX228,1 enables the Ctrl-Shift function keys, using OSCLI to program the selected key with the string returned by FNdate.

USING THE ROUTINE WITH VIEW

Enter the program and save as DateProg before trying it out, as the program will be lost as soon as View is entered.

Then prepare a !BOOT file by typing:

```
*Build $.!BOOT
```

and enter the lines:

```
1 *BASIC
2 CHAIN "$.DateProg"
3 MODE131
4 *WORD
```

Press Escape and enter:

```
*OPT4,3
```

and your disc is ready. Note the use of a shadow mode (by adding 128 to the mode number) to maximise the amount of memory available to View. Pressing Shift/Break will then boot the disc, automatically priming Ctrl-Shift-f6 with the current date. Once View has been entered, pressing Ctrl-Shift-f6 at any time will insert the current date into the text at the cursor position. If burning the midnight oil, remember to re-boot the program to change the date!

RE-SETTING THE CLOCK

If your Master does not show the correct time, you can readily change this from within Basic. Just type:

```
PRINT TIME$
TIME$="Day, dd Mmm yyyy.hh:mm:ss"
```

where the string assigned to TIME$ matches the format of that displayed by the PRINT instruction, for example:

```
"Sun, 05 Feb 1989.19:11:00"
```

Keying Return exactly on the minute will reset the clock to the required time.

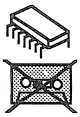
```
  10 REM Program DateProg
  14 th April, 1989
  20 REM Version 1.1
  30 REM Author  Jeff Gorman
  40 REM BEEBUG  May 1989
  50 REM Program subject to copyright
  60 :
 100 MODE 131
 110 ON ERROR GOTO 160
 120 *FX228,1
 130 OSCLI("Key6 "+FNdate)
 140 END
 150 :
 160 REPORT:PRINT " at line ";ERL
 170 END
 180 :
1000 DEF FNdate
1010 date$=MID$(TIME$,5,2)
1020 date=VAL(date$):date$=STR$(date)
1030 suf1$="th"
1040 IF date=1 OR date=21 OR date=31 su
f1$="st"
1050 IF date=2 OR date=22 suf1$="nd"
1060 IF date=3 OR date=23 suf1$="rd"
1070 date$=date$+suf1$+" "
1080 :
1090 mth$=MID$(TIME$,8,3)
1100 mth%=0
1110 REPEAT
1120 mth%=mth%+1
1130 READ month$
1140 UNTIL mth$=LEFT$(month$,3)
1150 mth$=month$+", "
1160 year$=MID$(TIME$,12,4)
1170 =date$+mth$+year$
1180 :
1190 DATA January,February,March,April,
May,June,July,August,September,October,
November,December
```

Big Text Screen Displays (continued from page 26)

```
1780 .ending
1790 JSR print
1800 .restore
1810 PLA:TAY:PLA:TAX
1820 RTS
1830 .print
1840 JMP(&FFCC)
1850 .vdu_codes \  check list of codes
1860 LDX#12
1870 .compare
1880 CMP codes,X:BEQ matched
1890 DEX:BPL compare
1900 BMI ending
1910 .matched
```

```
1920 LDY extra,X:BEQ up_down
1930 STY flag:BNE ending
1940 .up_down
1950 JSR print
1960 JMP ending
1970 .data_buffer
1980 ]
1990 NEXT
2000 ENDPROC
2010 :
2020 DEFPROCunbig
2030 ?&20E=?&FFCC
2040 ?&20F=?&FFCD
2050 ENDPROC
```

Transparent Sideways RAM Loader



Simplify the loading and saving of ROM images with this utility from Keith Harding.

On a model B with sideways RAM fitted, the process of loading a ROM image into the RAM is often awkward and inconvenient. In many cases a loader program has to be loaded from disc, and this then loads the image into main memory and copies it into sideways RAM. Thus any program in main memory is lost, and you have to have a copy of the loader to hand. The process of saving a ROM image back to a file can be even more convoluted!

The program presented here solves these problems by creating a utility which will load or save a ROM image without corrupting the contents of main memory. The loader utility is written in machine code, and there are two versions. The first of these is itself a ROM image, and once loaded into sideways RAM can be used to load and save other images. The second version resides in main memory, and is only really necessary when there is no free bank of sideways RAM to hold the loader, or perhaps to load the ROM version of the utility. Alternatively, the ROM version can be programmed into an EPROM and installed permanently.

To use the program, type in and save the listing given here. When you run it you are prompted to press 'M' or 'R' according to whether you want to assemble the main memory or ROM image version. Once assembled, the program will prompt for a filename, and will then save the machine code with this name. The *SAVE command is printed out for reference.

As it stands, the main memory version is assembled to run from address &1400. This is fine for use with DFS, but for ADFS you might need to assemble it to run elsewhere. This can be done by changing the &1400 in line 1010.

USING THE UTILITY

Before using the loader, it must itself be loaded. For the ROM image version, either load it using the method you normally use to load ROM images, or alternatively use the main memory version of the utility to load it. To load the main memory version simply use *<filename>. The sideways RAM loader provides two commands:

```
*SRLOAD <Rom No.> <Filename>
```
For example: `*SRLOAD 4 BASIC1`

This command loads the specified file into the specified ROM socket and, if it is a valid ROM initialises it. Initialising involves transferring the ROM type byte to the ROM table. It does not actually cause the ROM to claim private workspace if it needs it. This does not normally cause problems with language ROMs, but with utility ROMs it is wise to press Ctrl-Break to ensure the image is initialised. If the image cannot be loaded, either because the socket does not contain RAM or because it is write-protected then a Transfer Error is produced.

```
*SRSAVE <Rom No.> <Filename>
```
For example: `*SRSAVE 9 DFS_COPY`

This command saves to file the ROM image in the given ROM socket under the specified filename. It automatically selects whether the ROM is an 8K or 16K version and saves it accordingly.

USE WITH A MASTER

Normally, it will not be necessary to use the sideways RAM loader with a Master, because equivalent commands are provided. However, you may wish to use it nonetheless for two possible reasons. Firstly, the syntax of the built in commands is more complex (but also more flexible) than those here, and secondly, the Master will not normally allow you to save the contents of a ROM (only sideways RAM). However, the names of the commands provided by this utility will clash with those built in to the Master. To get round this you will need to change the command names in lines 3280, 3290, 4340 and 4380.

HOW IT WORKS

The sideways RAM loader works by transferring the data between the RAM and a file in 256-byte chunks. The file is opened for byte-based access, and the data transfer is achieved using the filing system call OSGBPB. Rather than transferring the data directly to and from the sideways RAM, a 256-byte buffer in main memory is used (at address &A00). This is because only one sideways ROM slot may be paged in at a time. Therefore, when the ROM containing DFS, ADFS etc. is active, the sideways RAM cannot be accessed directly. For the same reason, the routine to transfer the buffer to the sideways RAM has to be copied into main memory when using the ROM image version of the loader.

```
  10 REM Program Sideways Ram Loader
  20 REM Version B1.01
  30 REM Author  Kevin Harding
  40 REM BEEBUG  May 1988
  50 REM Program subject to copyright
  60 :
 100 MODE7:HIMEM=&7400
 110 PRINT"ASSEMBLE FOR ROM OR MEMORY (
R/M)? : ";
 120 REPEAT q$=CHR$(GET AND &DF)
 130 UNTIL q$="R" OR q$="M"
 140 PRINTq$'
 150 FORx=1 TO 2
 160 PRINTCHR$141;SPC(10);"ASSEMBLING T
O";CHR$136;
 170 IF q$="M" PRINT"MEMORY" ELSE PRINT
"ROM"
 180 NEXT:VDU28,0,24,39,5
 190 zp=&A8:cliv=&208:other=&380
 200 commandinput=zp:rampointer=zp
 210 filenamepointer=zp:emessage=zp
 220 commandpointer=zp+2
 230 routineaddress=zp+2
 240 rominfotable=zp+2:clivorg=other
 250 whichcommand=other+2
 260 romaddress=other+2
 270 atemp=other+3:xtemp=other+3
 280 romnumber=other+4:drive=other+5
 290 directory=other+6:sector=other+7
 300 track=other+8:lastchar=other+9
 310 initiaterom=other+9
 320 dircharvalid=other+10
 330 fnamecharvalid=other+11
 340 noofsectors=other+12
 350 startsector=other+13
 360 buffer=other+15:param=other+15
 370 filename=other+33
 380 name=other+47:transferbuffer=&A00
 390 transferprogram=&900
 400 PROCassemble
 410 INPUT "Filename ";name$
 420 A$="SAVE "+name$+" "+STR$~L%+" ":
 430 IFF%<4 A$=A$+STR$~P% ELSE A$=A$+ST
R$~(P%-&8000+L%)
 440 PRINT A$:OSCLI A$:END
 450 :
1000 DEFPROCassemble
1010 IFq$="R" THEN F%=4:L%=&7400 ELSE F
%=0:L%=&1400
1020 FORI%=F% TO F%+3 STEP 3
1030 IFq$="R" PROCrombreak:GOTO 1190
1040 P%=L%
1050 [OPTI%
1060 .setup LDY#0:LDX#&4C:LDA#247
1070 JSR &FFF4:LDY#0
1080 LDX#breakcontrol MOD 256:LDA#248
1090 JSR &FFF4:LDY#0
1100 LDX#breakcontrol DIV 256:LDA#249
1110 JSR &FFF4
1120 .setvectors LDA cliv:STA clivorg
1130 LDA cliv+1:STA clivorg+1
1140 LDA#starcommand MOD 256:STA cliv
1150 LDA#starcommand DIV 256:STA cliv+1
1160 RTS
1170 .breakcontrol:BCC dontset
1180 JSR setvectors:.dontset RTS:]
1190 IFq$="R" PROCromstarcommand:GOTO12
30
1200 [OPTI%
1210 .starcommand STX commandinput
1220 STY commandinput+1:]
1230 [OPTI%
1240 LDA#actualcommands MOD 256
1250 STA commandpointer
1260 LDA#actualcommands DIV 256
1270 STA commandpointer+1:LDY#&FF
1280 .searchfornonstar INY
1290 LDA (commandinput),Y:CMP#ASC"*"
1300 BEQ searchfornonstar:CLC:TYA
1310 ADC commandinput:STA commandinput
1320 BCC nocarrytoadd
1330 INC commandinput+1
1340 .nocarrytoadd LDX#&FE
1350 .comparenxtcommand LDY#0
1360 .comparenxtstarchar LDA (commandpo
inter),Y:BEQ commandmatched
1370 LDA (commandinput),Y
1380 JSR converttocapitals
1390 .comparecommandletter CMP (command
pointer),Y:BNE commandunmatched
```

```
 1400 INY:JMP comparenxtstarchar
 1410 .commandunmatched LDA (commandinpu
t),Y:CMP#ASC".":BNE findnxtcommand
 1420 INY:CPY#4:BCS commandmatched
 1430 .findnxtcommand CLC
 1440 LDA commandpointer:]
 1450 IFq$="R" [OPTI%:ADC#28:]:GOTO1480
 1460 [OPTI%
 1470 ADC#7:]
 1480 [OPTI%
 1490 STA commandpointer:BCC addifreq
 1500 INC commandpointer+1
 1510 .addifreq INX
 1520 BNE comparenxtcommand:]
 1530 IFq$="R" PROCromstarexit:GOTO1570
 1540 [OPTI%
 1550 .passoncommand LDX commandinput
 1560 LDY commandinput+1:JMP (clivorg):]
 1570 [OPTI%
 1580 .commandmatched STX whichcommand:]
 1590 IFq$="R" [OPTI%:PLA:PLA:PLA:]
 1600 [OPTI%
 1610 .checknxtinputchar LDA (commandinp
ut),Y:CMP#32
 1620 BNE foundstartofromnumber:INY
 1630 JMP checknxtinputchar
 1640 .foundstartofromnumber LDX#0
 1650 .fetchnxtdigit
 1660 LDA (commandinput),Y:CMP#32
 1670 BEQ finishednumber:CMP#13
 1680 BEQ syntaxerror:STA buffer,X:INX
 1690 INY:CPX#3:BNE fetchnxtdigit
 1700 .syntaxerror:]
 1710 IFq$="R" [OPTI%:JSR writeerrormess
age:]
 1720 [OPTI%
 1730 BRK:EQUB &62:EQUS "Syntax Error"
 1740 EQUB &00
 1750 .finishednumber DEX:LDA buffer,X
 1760 CMP#48:BCC syntaxerror:CMP#58
 1770 BCS syntaxerror:SEC:SBC#48
 1780 STA romnumber:DEX
 1790 BMI foundromnumber:LDA buffer,X
 1800 CMP#48:BEQ foundromnumber:CMP#49
 1810 BNE syntaxerror:CLC:LDA romnumber
 1820 ADC#10:CMP#16:BCS syntaxerror
 1830 STA romnumber
 1840 .foundromnumber LDX#13
 1850 STX lastchar:LDX#0
 1860 .findfirstfilenamechar INY
 1870 LDA (commandinput),Y:CMP#32
 1880 BEQ findfirstfilenamechar:DEY
 1890 .copynxtfilenamechar INY
 1900 LDA (commandinput),Y:DEY
 1910 CMP#ASC"""":BNE copyallfilename
 1920 STA lastchar:INY
 1930 .copyallfilename INY
 1940 LDA (commandinput),Y:CMP lastchar
 1950 BEQ foundendoffilename:CMP#13
 1960 BEQ syntaxerror:CMP#32
 1970 BEQ foundfnspace:STA filename,X
 1980 INX:CPX#13:BNE copyallfilename
 1990 BEQ syntaxerror
 2000 .foundfnspace LDA lastchar:CMP#13
 2010 BNE syntaxerror
 2020 .foundendoffilename LDA#13
 2030 STA filename,X
 2040 .finishedcopyacross
 2050 LDX whichcommand:INX
 2060 BNE sidewaysramload
 2070 JMP sidewaysramsave
 2080 .sidewaysramload LDA #&40
 2090 LDX #filename MOD &100
 2100 LDY #filename DIV &100
 2110 JSR &FFCE:STA param
 2120 CMP #0:BNE foundok:]
 2130 IFq$="R" [OPTI%:JSR writeerrormess
age:]
 2140 [OPTI%
 2150 BRK:EQUB &63:EQUS "No such file"
 2160 EQUB &00
 2170 .foundok:]
 2180 IF q$="R" [OPTI%:LDX#transpagetora
m MOD 256:LDY#transpagetoram DIV 256:JSR
 transferroutinetoram:]
 2190 [OPTI%
 2200 LDA#0:STA rampointer
 2210 JSR initactrom:LDA#&80
 2220 STA rampointer+1
 2230 .transfernxtloadsector LDA #0
 2240 STA param+5
 2250 STA param+7:STA param+8
 2260 LDA #&FF:STA param+3:STA param+4
 2270 LDA #transferbuffer DIV 256
 2280 STA param+2
 2290 LDA #transferbuffer MOD 256
 2300 STA param+1
 2310 LDA #1:STA param+6:LDA #4
 2320 LDX #param MOD 256
 2330 LDY #param DIV 256:JSR &FFD1:PHP
 2340 LDA param+6:BNE nonedone
 2350 JSR putpageinrom:INC rampointer+1
 2360 LDA#ASC".":JSR&FFE3
 2370 .nonedone
 2380 PLP:BCC transfernxtloadsector
 2390 LDA #0:LDY param:JSR &FFCE
 2400 JSR&FFE7:LDA#7:STA romaddress
 2410 LDA#&80:JSR fetchvaluefromrom
```

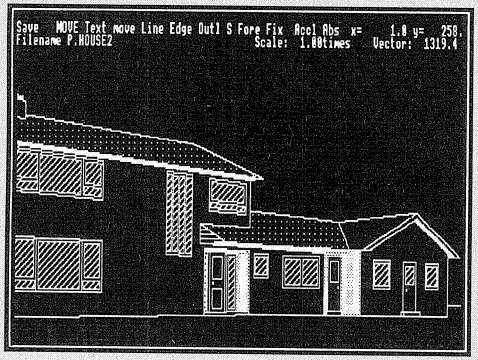
```
2420 STA romaddress:LDX#0
2430 .checknxtrcbyte STX xtemp:LDA#&80
2440 JSR fetchvaluefromrom:LDX xtemp
2450 CMP crightstart,X
2460 BNE donotinitialise:INC romaddress
2470 INX:CPX#4:BNE checknxtrcbyte
2480 LDA#6:STA romaddress:LDA#&80
2490 JSR fetchvaluefromrom
2500 JSR initactrom
2510 .donotinitialise LDA#0:RTS
2520 .initactrom PHA:LDA#170:LDX#0
2530 LDY#&FF:JSR&FFF4:STX rominfotable
2540 STY rominfotable+1:LDY romnumber
2550 PLA:STA (rominfotable),Y:RTS
2560 .crightstart EQUB &00:EQUS "(C)"
2570 .transpagetoram:]
2580 IFq$="R" K%=P%:P%=transferprogram
2590 [OPTI%
2600 .putpageinrom LDY romnumber:LDA#0
2610 STY&FE30:STA &FF30,Y:TAY
2620 .transfernxtbyte
2630 LDA transferbuffer,Y
2640 STA (rampointer),Y
2650 CMP (rampointer),Y
2660 BNE transfererror:INY
2670 BNE transfernxtbyte
2680 .resetrompointer LDY&F4:STY &FE30
2690 LDY#0:STY&FF38:RTS
2700 .transfererror JSR resetrompointer
2710 LDA #0:LDY param:JSR &FFCE
2720 BRK:EQUB &61
2730 EQUS "Transfer Failed":EQUW &0000
2740 ]
2750 IFq$="R" P%=P%-transferprogram+K%:
PROCtransroutinetoram
2760 [OPTI%
2770 .sidewaysramsave LDA#0
2780 STA romaddress
2790 .comparenxt8kbyte LDA#&80
2800 JSR fetchvaluefromrom:STA atemp
2810 LDA#&A0:JSR fetchvaluefromrom
2820 CMP atemp:BNE setfor16ksave
2830 INC romaddress
2840 BNE comparenxt8kbyte:LDA#&20
2850 JMP writeblankfile
2860 .fetchvaluefromrom STA &F7
2870 LDA romaddress:STA &F6
2880 LDY romnumber:JSR &FFB9:RTS
2890 .setfor16ksave LDA#&40
2900 .writeblankfile:STA noofsectors
2910 LDA #&80:LDX #filename MOD 256
2920 LDY #filename DIV 256
2930 JSR &FFCE:STA param:]
2940 IFq$="R" [OPTI%:LDX#transpagefromr
am MOD 256:LDY#transpagefromram DIV 256:
JSR transferroutinetoram:]
2950 [OPTI%
2960 LDA#0:STA rampointer:LDA#&80
2970 STA rampointer+1
2980 .transfernxtsavesector
2990 JSR pullpagefromrom:LDA#&4B
3000 LDA #0:STA param+5:STA param+6
3010 STA param+7:STA param+8
3020 INC param+6:LDA #&FF
3030 STA param+3:STA param+4
3040 LDA #transferbuffer MOD &100
3050 STA param+1
3060 LDA #transferbuffer DIV &100
3070 STA param+2
3080 LDA #2:LDX #param MOD &100
3090 LDY #param DIV &100:JSR &FFD1
3100 LDA#ASC".":JSR &FFE3
3110 INC rampointer+1:DEC noofsectors
3120 BNE transfernxtsavesector
3130 LDA #0:LDY param:JSR &FFCE
3140 JSR&FFE7:LDA#0:RTS
3150 .transpagefromram:]
3160 IFq$="R" K%=P%:P%=transferprogram
3170 [OPTI%
3180 .pullpagefromrom LDA romnumber
3190 STA &FE30:LDY#0
3200 .transferbacknxtbyte
3210 LDA (rampointer),Y
3220 STA transferbuffer,Y:INY
3230 BNE transferbacknxtbyte:LDA&F4
3240 STA &FE30:RTS:EQUW &0000:]
3250 IFq$="R" P%=P%-transferprogram+K%
3260 IFq$="R" PROCcommands:GOTO3300
3270 [OPTI%
3280 .actualcommands EQUS "SRLOAD"
3290 EQUB &00:EQUS "SRSAVE":EQUB &00:]
3300 [OPTI%
3310 .converttocapitals CMP#ASC"a"
3320 BCC charok:CMP#ASC"z"+1:BCS charok
3330 AND#&DF:.charok RTS:]
3340 IFq$="R" PROCromerror
3350 NEXT
3360 ENDPROC
3370 :
3380 DEFPROCromerror
3390 [OPTI%
3400 .writeerrormessage PLA
3410 STA emessage:PLA:STA emessage+1
3420 LDA#0:STA&100:LDY#1
3430 .copyerrormessage INY
3440 LDA (emessage),Y:STA &FF,Y
3450 BNE copyerrormessage:JMP &100:]
3460 ENDPROC
```

```
3470 :
3480 DEFPROCrombreak
3490 P%=&8000
3500 O%=L%
3510 [OPTI%
3520 EQUB &00:EQUW &0000
3530 JMP serviceentry
3540 EQUB &82
3550 EQUB copyrightpointer MOD 256
3560 EQUB &01
3570 EQUS "SIDEWAYS RAM LOADER"
3580 EQUB &00
3590 EQUS "1.01"
3600 .copyrightpointer EQUB &00
3610 EQUS "(C) BEEBUG 1989":EQUB &00
3620 .serviceentry CMP#4
3630 BNE checkforhelp:JMP starcommand
3640 .checkforhelp CMP#9
3650 BEQ helpcommand:RTS
3660 .helpcommand PHA:TYA:PHA:TXA:PHA
3670 DEY
3680 .searchforhelpcom INY:LDA(&F2),Y
3690 CMP#32:BEQ searchforhelpcom
3700 CMP#13:BNE acheckforcommand
3710 JSR writeoutname:LDY#0
3720 .writenxtcomchar LDA command-2,Y
3730 BEQ comwordwritten:JSR &FFE3:INY
3740 BNE writenxtcomchar
3750 .comwordwritten JSR &FFE7
3760 JMP rregisters
3770 EQUW &2020
3780 .command EQUS "COMMANDS":EQUB &00
3790 .acheckforcommand LDX#0
3800 .comparenxtcommandchar LDA (&F2),Y
3810 JSR converttocapitals:CMP#13
3820 BEQ commandend:CMP#32
3830 BEQ commandend:CMP command,X
3840 BNE checkfordot:INY:INX
3850 BNE comparenxtcommandchar
3860 .checkfordot CMP#ASC"."
3870 BNE acommandnotmatched:CPX#2
3880 BCS acommandmatched
3890 .acommandnotmatched JMP rregisters
3900 .commandend LDA command,X
3910 BNE acommandnotmatched
3920 .acommandmatched JSR writeoutname
3930 LDY#0
3940 .writenxtcom LDA#32:JSR&FFE3
3950 JSR&FFE3
3960 .writenxtcomletter
3970 LDA actualcommands,Y
3980 BEQ writesyntax:JSR&FFE3:INY
3990 JMP writenxtcomletter
4000 .writesyntax LDA#32:JSR&FFE3
4010 .writenxtsynchar INY
4020 LDA actualcommands,Y
4030 BEQ syntaxwritten:JSR &FFE3
4040 JMP writenxtsynchar
4050 .syntaxwritten JSR &FFE7:INY
4060 LDA actualcommands,Y
4070 BEQ rregisters:BNE writenxtcom
4080 .rregisters PLA:TAX:PLA:TAY:PLA
4090 RTS
4100 .writeoutname JSR&FFE7:LDY#8
4110 .writenxttitlechar INY:LDA&8000,Y
4120 BEQ finishedtitlestring:JSR&FFE3
4130 JMP writenxttitlechar
4140 .finishedtitlestring LDA#32
4150 JSR&FFE3:CPY&8007
4160 BNE writenxttitlechar:JMP &FFE7:]
4170 ENDPROC
4180 :
4190 DEFPROCromstarcommand
4200 [OPTI%
4210 .starcommand PHA:TYA:PHA:TXA:PHA
4220 TYA:CLC:ADC &F2:STA commandinput
4230 LDA &F3:ADC#0:STA commandinput+1:]
4240 ENDPROC
4250 :
4260 DEFPROCromstarexit
4270 [OPTI%
4280 .passoncommand PLA:TAX:PLA:TAY:PLA
4290 RTS:]
4300 ENDPROC
4310 :
4320 DEFPROCcommands
4330 [OPTI%
4340 .actualcommands EQUS "SRLOAD"
4350 EQUB &00
4360 EQUS "<Rom No.> <Filename>"
4370 EQUB &00
4380 EQUS "SRSAVE":EQUB &00
4390 EQUS "<Rom No.> <Filename>"
4400 EQUB &00:EQUB &00:]
4410 ENDPROC
4420 :
4430 DEFPROCtransroutinetoram
4440 [OPTI%
4450 .transferroutinetoram STX routinea
ddress:STY routineaddress+1
4460 LDY#&FF
4470 .transnxtbytetoram LDX#0
4480 .foundfirsttranszero INY
4490 LDA (routineaddress),Y
4500 STA transferprogram,Y:CMP#0
4510 BNE transnxtbytetoram:INX:CPX#2
4520 BNE foundfirsttranszero:RTS:]
4530 ENDPROC
```

ASTAAD

The Best of BEEBUG

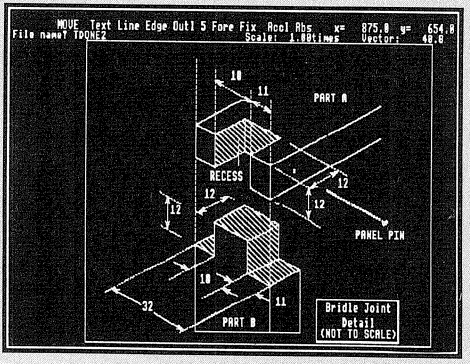
Our new BEST OF BEEBUG disc is based on the recent series of articles and programs dealing with ASTAAD, our comprehensive CAD program for the Master and Compact. The ASTAAD disc contains everything you need to use this very impressive piece of software, including the enhanced versions of the ASTAAD and STEAMS programs. This is supplied complete with documentation and printed keystrips for use with ASTAAD and STEAMS.



ASTAAD £9.95

* Enhanced ASTAAD CAD program now including full mouse and joystick control, built-in printer dump and speed improvement.
* STEAMS image manipulator
* 8 page manual covering the use of the system.
* Keystrips for ASTAAD and STEAMS
* Sample picture files to experiment with.

Recent magazines have shown many examples of the truly impressive results that can be achieved with ASTAAD for producing technical drawings and diagrams including perspective work. The disc also contains several example drawings for you to load and modify, for easy familiarisation with the software. All this is at a fraction of the price that you would need to pay for equivalent commercial software.



Our first two releases under the Best of BEEBUG label are still available:

General Utilities £5.75
Best of BEEBUG Disc 1

* Printer Buffer * * ROM Controller * Sprite Editor/Animator * Multi-Character Printer Driver for View * Mode 7 Screen Editor * Multi-Column Printing * Epson Character Definer * BEEBUG MiniWimp † * ROM Filing System Generator

*** Master series only. † Requires sideways RAM.**

Applications £5.75
Best of BEEBUG Disc 2

* Business Graphics * Video Cataloguer
* World by Night and Day * Phone Book
* Page Designer * Personalised Letter-Heads
* Mapping the British Isles * Selective Breeding
* Appointments Diary * The Earth from Space
* Personalised Address Book

Please rush me my Best of BEEBUG discs:

ASTAAD Code 1407A *(80 track DFS)* ❑
Code 1408A *(3.5" ADFS)* ❑

General Utilities Disc Code 1405A ❑
Applications Disc Code 1404A ❑

Name _____

Address _____

_____

Memership No _____

Total £_____
Postage (£0.60 + £0.30 for every other disc) £_____
Grand Total £_____

I enclose a cheque for £_____ OR please debit my Access, Visa or Connect account, Card No_____/_____/_____/_____ Expiry____/____ Signed_____

Return to BEEBUG Ltd, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX. Telephone (0727) 40303.

1st course — Investigating Teletext Mode (2)

Mike Williams extends beyond the double-height characters described last month to cover techniques for even larger mode 7 displays.

What if we want to display digits in a size larger than can be achieved with double height text? Well there is a solution, and that is to design our own digits and use teletext mode graphics to display these at whatever size we want.

SINGLE DIGIT DISPLAY

However, lets keep everything as simple as possible. The mode 7 character definitions used by Basic are all listed in the back of your User Guide. If you look at these, you will see that they are based on a matrix of 7 rows by 5 columns. We will use exactly the same definitions, but represent each pixel by the single mode 7 character CHR$255 (a solid rectangle).

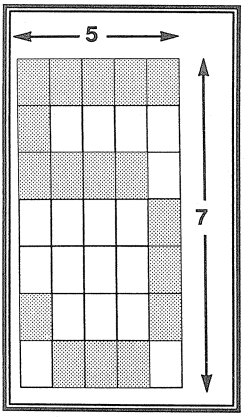


Figure 1. Design of digit '5'

Thus to display the top-most line of the digit '5' (see Fig 1), we could write:

```
PRINT CHR$(255);CHR$(255);CHR$(255);
  CHR$(255);CHR$(255)
```

Now this may begin to look quite cumbersome, but there is a much simpler way using VDU rather than PRINT (see previous First Course articles for more information on the relative merits of these two statements). Our previous statement then becomes:

```
VDU255,255,255,255,255
```

In fact the best way of proceeding is to write a complete procedure which will display a large figure five on the screen in the position (x,y) and graphics colour (g) of our choice:

```
1100 DEF PROCfive(x,y,g)
1110 LOCAL I%
1120 FOR I%=1 TO 7
1130 READ a1,a2,a3,a4,a5
1140 PRINTTAB(x,y+I%);
1150 VDU g,a1,a2,a3,a4,a5
1160 NEXT I%
1170 ENDPROC
1200 DATA 255,255,255,255,255
1210 DATA 255,32,32,32,32
1220 DATA 255,255,255,255,32
1230 DATA 32,32,32,32,255
1240 DATA 32,32,32,32,255
1250 DATA 255,32,32,32,255
1260 DATA 32,255,255,255,32
```

Each line of data corresponds to one line of our 5 by 7 digit, and comprises either the graphics character 255, or a space (code 32). Each line of data is read in turn, and displayed using VDU in line 1150

With this definition try typing:

```
PROCfive(18,10,147)
```

to see a yellow '5' displayed on the screen. Similarly, procedures could be written for the other nine digits. The question which arises is how we can apply such procedures to the display of a whole number. Furthermore, it seems rather cumbersome to use a different procedure for each digit. Surely there must be some way of writing a general purpose procedure which will handle all digits.

We will assume that any number to be displayed in this way is converted into a corresponding string (using STR$), and as well as the digits 0 to 9 we might as well include the minus sign (same as a hyphen) and a decimal point (full stop) so that we are not unduly limited as to the numbers we can display. It will also be useful to include a space character for reasons which will become clear later. That will make 13 characters in all that we must cope with.

It could be done, by extracting each digit of a given number in turn, and by the use of IF statements calling a corresponding procedure. But it would be very cumbersome, and there is a much better way. Several ideas are involved and we will take these one at a time.

In the procedure above we needed a PRINT TAB statement to ensure that each row of graphic characters was positioned in the right place. We will now abandon that idea, and instead we will define a text window with its top left-hand corner in the specified position. Ideally, we need a window 5 characters wide and 7 characters high, but in practice that causes problems, as any character displayed in the bottom right-hand corner of the window forces the window contents to scroll. We shall therefore define a window (5 wide by 8 high) with one extra row at its foot to avoid this:

```
VDU28,x,y+7,x+4,y
```

The reason why we need x+4 and y+7 (rather than adding 5 and 8) should be clear from Figure 2. The loop from 1120 to 1160 can now be replaced as follows:

```
1120 VDU28,x,y+7,x+4,y
1130 FOR I%=1 TO 35
1140 READ a:VDU a
1150 NEXT I%
```

Change the procedure and call it as before to see that it still works. The new coding is much simpler. Each character is read one at a time and 'poured' into the window. I must confess that we have temporarily lost any colour, but we will rectify that later.

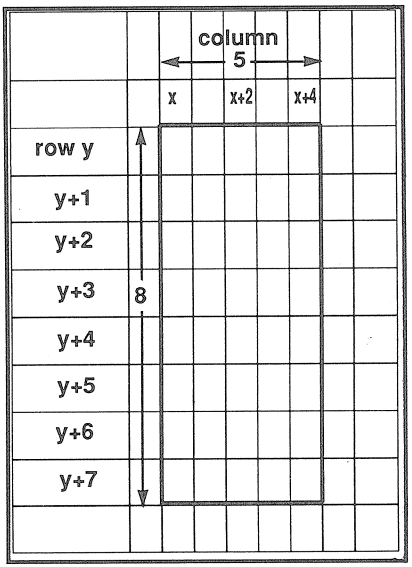


Figure 2. Defining a digit window

GENERAL DIGIT DISPLAY

We can now use this technique to convert our procedure, which will only display a '5', into a more general purpose routine which will deal with any digit. Let us assume that the characters defining all the digits are specified as a series of data statements, with all the digits defined in order from 0 to 9, followed by '.' and then '-'. We don't need to define 'space' in this way, fortunately, but use a much simpler technique (see later). We will carefully number the DATA statements so that the definition of '0' starts at line 9000, that for '1' at line 9100, and so on. We can then use a RESTORE statement to position the DATA pointer at the start of the definition for a specified digit.

If we want to display digit 'n'. We could simply write the RESTORE statement as:

```
RESTORE 9000+100*n
```

and the DATA pointer will be moved to the correct position. The only difficulty is with the '.', '-' and 'space'. There are various ways of solving this, but we will use INSTR.

Thus given a digit as a character, digit$, we will use INSTR to determine which definition is required:

```
n=INSTR("0123456789.- "
  ,digit$)
RESTORE 8900+100*n
```

Note the adjustment to the RESTORE statement as the '0' now appears in position '1', and so on. This has the added advantage that n with a value of '0' will indicate an unrecognised digit. Note, too, the 'space' at the end of the character string in the INSTR statement.

We will now completely rewrite the original procedure as follows to provide a general purpose digit display.

```
1100 DEF PROCdigit(x,y,digit$)
1110 LOCAL n,I%
1120 n=INSTR("0123456789.- ",digit$)
1130 IF n=0 ENDPROC
1140 VDU28,x,y+7,x+4,y
1150 IF n=13 CLS:ENDPROC
1160 RESTORE (8900+100*n)
1170 FOR I%=1 TO 35
1180 READ a:VDU a
1190 NEXT I%:VDU26
1200 ENDPROC
```

Note the use of CLS in line 1150 to clear the digit window, and thus implement a 'space' when needed. You can try this out with your data for the figure '5', but make sure that the DATA statements are renumbered from 9500.

Build up a complete set of DATA statements, and you should find that the new procedure will display any digit (or a '.', '-' or a 'space'). I know that means 420 numbers in all, which is a bit tedious, but it does keep things relatively simple for now.

DISPLAYING A NUMBER

Now that we have a general procedure for displaying any digit, let's see how we can apply that to the display of a complete number. We need to know in advance the maximum number of digits, including any minus sign and decimal point, which a display will need to cope with. In any case, as each large digit uses rows of five characters, the maximum number of digits displayable on a mode 7 screen is 8 (5 * 8 = 40). But that leaves no separation between digits. If we allow one space character between each digit, the maximum number of digits possible is now 6 (effectively each digit, including the space, is now 6 characters wide, and 6 * 6 = 36). Figure 3 shows how a four digit number would be positioned centrally on the screen.

In order to display a number we will write a new procedure (PROCnumber) with two parameters, the number to be displayed (n), and the maximum number of digits (d) to be used. The coding is as shown.

```
1000 DEF PROCnumber(n,d)
1010 LOCAL d$,n$,I%,L%,x,y:y=8
1020 n$=STR$(n):L%=LEN(n$)
1030 IF d>L% n$=STRING$(d-L%," ")+n$
1040 FOR I%=n TO 1 STEP -1
1050 d$=MID$(n$,I%,1)
1060 x=20-3*d+6*(I%-1)
1070 PROCdigit(x,y,d$)
1080 NEXT I%
1090 ENDPROC
```

Add this procedure to PROCdigit and your DATA statements, and then add some additional lines such as:

```
100 MODE7
110 PROCnumber(23.2,5)
120 END
```

to form a complete program. This short program will display the number 23.2 assuming a maximum 5 digit display.

The procedure converts the given number into string form (line 1020), and then uses a FOR-NEXT loop to extract each digit (character) in turn from right to left (line 1050). Depending upon which digit is to be displayed, line 1060 determines the x position (column) for the start of that digit's display, and PROCdigit is called to put the digit on the screen.

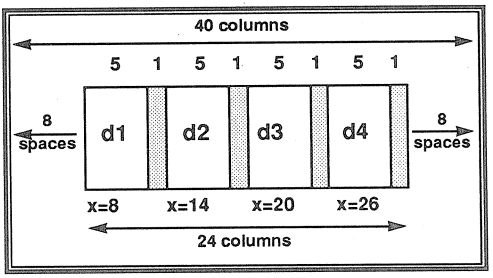


Figure 3. Positioning of a 4 digit display

You may wonder how on earth the formula in line 1060 is derived. Clear and careful thinking is the answer - see the explanation at the end of this article.

But what about colour you may still be asking. Well that is simple. However many digits we use, up to the maximum of 6, there will be unused character positions at the start of each row on which digits are displayed. So we just put a column of teletext colour codes down the side of the screen, and all the digits will be in the specified colour.

To implement this, change the procedure definition to:

```
DEF PROCnumber(n,d,g)
```

where 'g' is the graphics colour code (145 to 151), and add three lines:

```
1012 FOR I%=0 TO 7
1014 PRINTTAB(0,I%+y)CHR$(g)
1016 NEXT I%
```

Then change line 110 to read:

```
110 PROCnumber(23.2,5,147)
```

to see the number displayed in yellow.

We can apply our procedures to the counting program developed last month (see the left-hand column on page 27). Simply replace lines 130 to 150 by the one line:

```
130 PROCnumber(T%,3,150)
```

and the procedure PROCdouble by this month's procedure definitions and DATA statements.

Continued on page 52

The Comms Spot

by Peter Rochford

In a previous Comms Spot (Vol.6 No.10) I boldly predicted that the price of high speed V22bis (2400/2400 baud) modems would fall dramatically with the introduction of BT's VASSCOM. Well, here we are a year later, and although the price of these modems has fallen, it has not been quite as dramatic as I expected, and there has not been a spate of cheap V22bis units unleashed on the market. Omelette sur le visage!

However, I did also mention in that article that Amstrad had produced a V22bis modem card for PC's, retailing at £199 plus VAT. I went on to suggest that, knowing Amstrad, it would not be too long before they produced a standalone version at a similar kind of price. Well, I was right about that anyway!

At £249 plus VAT, about half the price of most of the current V22bis competitors, the new Amstrad SM2400 modem is still not what you could call cheap, but certainly is by far the cheapest of its type.

The SM2400 is a BABT approved V21, V23, V22, and V22bis modem that is Hayes command compatible. It features auto-dial, auto-answer and has auto baud rate detection. It should work with any computer that has an RS232 interface, which includes of course both the BBC Micro and the Archimedes. Of course you will need suitable software, such as BEEBUG's own Hayes Command ROM for the Beeb or Hearsay for the Arc.

Unlike many other Hayes modems around, the SM2400 does not feature a built-in number store and battery-backup, but this is not such a problem, as many comms packages now feature their own database directory and will allow selection and auto dial.

Product	**Amstrad SM2400 Modem**
Supplier	**Amstrad Plc** **Brentwood House, 169 Kings Road,** **Brentwood, Essex CM14 4EF.** **Tel. (0277) 262326**
Price	**£286.35 inc. VAT**

The modem is housed in a beige plastic case, no more than 1 inch high, about 6 inches wide and 10 inches deep. It is a rather attractive nicely-styled unit. The power supply is external and remarkably large for a modem. Attached to it is a flying lead that supplies the 9.5V output to the modem via a socket at the rear.

Also at the rear of the modem is a 25-way D connector for hooking up to the computer's RS232 serial port. Alongside this is a telephone socket for connection of the displaced telephone.

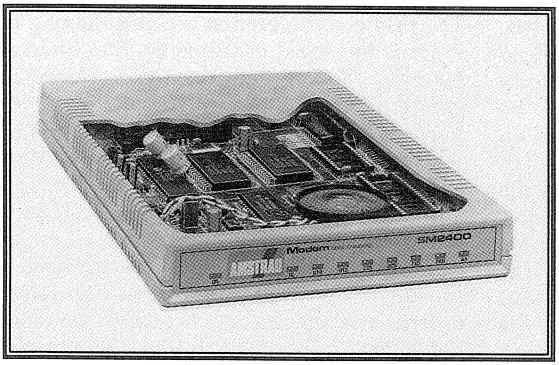
The front panel supports no less than nine LEDs. Six of these are for monitoring the line status of DTR, RTS, CTS, DCD, TXD and RXD. The other three indicate power, on-line and auto-answer. There are no controls or switches, everything is handled from the computer via software.

I was surprised to find the lack of a reset button on the rear of the modem. My own V22 Hayes modem has one and gets used fairly frequently to escape from a situation where I cannot force the modem offline by software control. You have no choice with the Amstrad but to disconnect the power supply. Awkward and annoying to say the least.

A connection lead was supplied with the review modem but was suitable only for use with a PC. It was therefore out with the trusty soldering iron and a hunt around the junk box

for the bits to make up a lead to work with the Master. Half-an-hour later with burnt fingers and frayed temper the job was complete!

With everything connected up, there was no trouble getting on-line. I used several comms packages on the Master to test the modem, including the BEEBUG Command ROM and Softmachinery's Commsoft. The latter by the way will not work at 2400 baud, a fault of the software, not the modem.



Amstrad SM2400 modem

I have waxed lyrically in this column about the joys of 2400 baud before, so I won't put you through that again. Suffice to say that the modem worked perfectly at that speed and at the lower 1200 and 300 baud rates.

The SM2400 has the peculiarity of not being able to communicate with the terminal at split baud rates (1200/75). As the modem always automatically switches itself to the baud rate of the terminal, to get the modem to operate at 1200/75 you have to send an AT command to it.

It is said that working at 2400 baud you need some kind of error correction. I found little to complain about whilst using the Amstrad at this speed. There is no built in MNP error correction on the modem, but you can't really complain at the price! For those using Prestel, if you have for example BEEBUG's Hearsay on the Arc, you can use the VASSCOM error correction facility in the software if need be.

DOCUMENTATION

The manual supplied with the modem is a rather skimpy affair of some forty pages. It starts off with a brave attempt to provide an introduction to comms for the newcomer. It is far too brief and, to my mind, not enlightening.

The rest of the manual is none too clever either and far from adequate. Technical data is supplied with little information on how to interpret it or make use of it. In some places, the technical descriptions of features are totally confusing and read as gobbledegook.

As you can gather, I think the documentation is grim and needs a complete re-write to make it clear and understandable.

CONCLUSION

Amstrad have the knack of coming up with a good consumer electronics product at an affordable price. This is true of the SM2400 modem. At this price there is nothing to touch it on the market.

It has an excellent specification apart from a few minor niggles and is built to a good standard. During the course of this review it performed faultlessly and in the main was a pleasure to use.

The only thing that marred the experience (apart from the burnt fingers!) was the poor documentation. This is a shame as it does let the product down.

If you need a high speed Hayes compatible modem and can shell out the asking price, you have a real bargain when compared to the competition. Definitely recommended. B

File Handling for All (Part 11)

David Spencer and Mike Williams take a look at some miscellaneous topics as our series on file handling draws near to a close.

INDEXING

If you want to find a particular topic in a reference book, you would not in general read through the book from page one until you found the desired bit. Instead, any sane person would refer to the index and use this to immediately identify the correct page. Exactly the same principle applies to databases. Given an index to the records, it will in general be much quicker to find a record than by having to search each one in turn. If the index is in a form which can be held in memory, then the searching can be even faster. The use of an index also offers other advantages. In particular, it means that the records within a file can be accessed in order without actually sorting the records. Furthermore, if separate indexes are set up for several key fields, it is possible to search the database without resorting to the main data file. Of course, the use of an index is not without cost. The penalty for the faster access is the extra storage needed to hold the index.

So what must our index hold? Considering again a book index, each entry consists of the word (or reference etc.) being indexed, and the page number of where to find it. Our file index will hold the same information - a key that will be used when searching the index, and a pointer to the corresponding record. The format of the key field will vary according to its contents, but will always be the same as the real field in the record it indexes. The format of the pointer field can also vary. For example, it could be an absolute value for the file pointer (one which could be set directly using PTR#), or it could be a record number, in which case it would need to be adjusted according to the record length to get the real pointer. It is common practice to keep the index in a separate file to the data records that it indexes, and the index will therefore normally also hold information about the file to which it refers.

Having explained the principle of an index, we now need to consider its implementation. We will not go into any great detail of how this done as the subject can get quite complex, and in any case the Beeb lacks the power, and memory, to handle a practical indexed database.

The prime consideration when designing any index is the speed of searching. If you were to take just the key field from each record and create a linear index, then searching for a particular field might well take as long as searching through the original records. To find a solution to the speed problem, consider again a printed index. Firstly, all the entries are sorted into order, and secondly, most indexes are split into sections for each letter of the alphabet. We can do exactly the same for our file index - order it and split it into chunks. In the case of fields containing character strings we might well split the records according to the first letter of each entry. However, the way we split the data is quite arbitrary, and will of course depend on the type of data. Additionally, we don't have to stop at just one split. If there are still a large number of entries in each group then they can be split again - perhaps according to the second letter of the string. This can continue until the groups are of such a size that they can easily be searched.

To give an indication of the efficiency of such a system, consider the case of an index of 6760 words. We will assume that the distribution of the first two letters of the words is uniform. In other words, no two letter prefix will occur more than another. If we were to search the list linearly, the average number of entries looked at would be 6760/2, or 3380. If instead we split the words into twenty-six lists according to the first letter, and then split each list again in the same way, then we will end up with 676 lists each containing just ten words. But it will now

take just two operations to locate the correct list, and an average of five operations to find the word in the list. We have therefore reduced the average number of search operations from 3380 to 7, a saving of 99.8%.

UPDATING THE INDEX

Obviously the index must be kept up to date, which in practice means revising it each time a record is added or deleted, or each time the key field of a record is changed. When a record is deleted, its entry in the index must also be removed. Similarly, when a record is added an index entry must be created for it and then placed in the correct position in the index. The situation of the key field of a record being changed is probably best handled by pretending that the record has been deleted, and a new one added, because this will not involve any extra code.

RELATED FILES

Another use of an index is to link a number of files together. A record in one file could include a key which could then be used to look up another record in the index to a different file. This could be extended indefinitely to create an entire set of interlinked files.

READING AND WRITING BLOCKS OF DATA

All our example routines given earlier in this series read and wrote files one byte at a time using the function BGET or BPUT (in some cases packaged up in the form of PRINT# and INPUT#). However, it is also possible to read or write a complete block of data in a single operation, with the user defining the length of the block. Typically, the length of the block would correspond to the record length, so that complete records could be read or written with a single command. As a further bonus, this also allows you to specify where in the file the data is to be located, saving a separate instruction to set PTR#.

The facility that allows us to read and write blocks is in the form of an operating system call with the name OSGBPB (Operating System Get Bytes - Put Bytes). There is no Basic keyword equivalent to OSGBPB, instead the routine must be called directly with the statement:

```
CALL &FFD1
```

Before making this call you must specify where in memory the data should be transferred to (or from), where the data is in the file, how many bytes to transfer, and which open file to use. All this information is passed to the routine in a *parameter block* which is simply an area of memory into which the various pieces of information are placed. For OSGBPB this parameter block is thirteen bytes long, and is arranged thus:

Byte 0	Channel number (handle)
Bytes 1-4	Address of data in memory
Bytes 5-8	Length of data to transfer
Bytes 9-12	Value of PTR to use

The first stage in setting up a parameter block is to allocate some memory for it. This can be done using the statement:

```
DIM blk 12
```

which will reserve thirteen bytes of memory, and store the start address of this block in the variable *blk*. The values can then be put into our block using the two *indirection operators* ? and ! which write a single byte and four bytes respectively. The code to set up the parameter block could look like the following:

```
?blk = channelnumber
blk!4 = buffer
blk!8 = length
blk!12 = position
```

The first line pokes in the channel number for the chosen file, which must have previously been opened using OPENIN etc. The next line sets the address in memory where the data is located, or where it is to be read to. We will come back to this in a moment. The third line sets the length of the block to be transferred, this value being in bytes (single characters). The final line sets the position in the file. This is the same as the value that would be given to the PTR# statement.

Once the OSGBPB parameter block has been set up, the routine can be called. Before executing the CALL statement, the variables A%, X% and Y% must be set to the operation number and the address of the parameter block. This latter is performed using:

```
X%=blk MOD &100
Y%=blk DIV &100
```

assuming that the variable pointing to the parameter block is *blk*, as in our example. The value in A% determines the exact operation which will be performed, and there are four which are of interest to us:

A%=1	Write to file using given pointer.
A%=2	Write to file using current pointer.
A%=3	Read bytes from file using given pointer.
A%=4	Read bytes from file using current pointer.

The meaning of the four commands should be fairly self-explanatory. You can either read from or write to a file, and you can either use the current value of the file pointer (PTR#), or use the value given in the parameter block. In the former case, the file pointer is updated once the transfer has been completed, while if the given position is used, then the file pointer is left unchanged.

Unlike BGET and BPUT, OSGBPB transfers the data to or from a block of memory. As for the parameter block, this is best reserved using DIM. In the above example we have assumed that the address of the block is stored in the variable *buffer*, and we should therefore reserve the block using a command such as:

```
DIM buffer maxlen
```

where *maxlen* is the length of the longest block that is likely to transferred.

Obviously the records must be put in, and read from the data block before the OSGBPB call is made. This is done using the indirection operators ?, ! and $ to read and write bytes, 4-bytes and strings. For example, a string could be put into the block with a command of the form:

```
$block=field$
```

You might be wondering why go to the hassle of reading and writing blocks rather than single bytes. The main reason is one of speed. The time taken to transfer a block of, say, 200 bytes will be approximately halved if the bytes are transferred as a single block, rather than one by one. In the case of an Econet system, the speed increase is much more dramatic - as much as a hundred times in some cases.

MULTIPLE FILE ACCESS

So far, our database model has consisted of one or more files which are read and updated as necessary. We have assumed, though not explicitly stated, that our access to the file is the only access being made at any one time. For a stand alone computer working with discs, this is a safe assumption, as only one program may run at a time. However, if the computer is connected to a network (Econet) then the entire situation changes because in theory any station on the network can access the file. Furthermore, as each computer is relatively independent, there is unlikely to be any synchronisation between file accesses, even if two computers are running exactly the same program.

So does this pose a problem? The answer is it depends on how the file is being accessed. Consider first the situation where an unknown number of stations on the network are reading records from the file. Each station will have had to open the file for input (using OPENIN probably), and the network file server (the device responsible for storing and handling files) will have allocated each station a separate channel with its own handle. Each station can move around the file at will (using PTR#), and read records. As the file contents are not being altered at all, each station is totally unaware that another is in fact also accessing the same file.

Consider now what happens if one station decides it wants to change a record in the file.

Making any changes to a file will potentially upset the entire file. For example, inserting one record near the start of the file might shuffle up all those after it. Any programs reading a particular file would have no way of telling that it had been changed. So alleviate this, and other related problems, the Beeb filing systems employ a technique called *multi read, single write*. As its name suggests, this allows a file to be open many times simultaneously for reading, but only once for writing.

The multi read, single write system is implemented with the following algorithm:

If a file is not open at all, then it may be opened for reading or writing.

If a file is open for reading only then it can be opened for reading, but not for writing.

If a file is open for writing, it cannot be opened for either reading or writing.

If an attempt to open a file is blocked by this system, then an error of the form 'Can't - File open' (ADFS), or 'Already open at station nnn' (Econet) will result. One point to note is that when a file is opened for update using OPENUP, it is implicitly opened for writing, and cannot therefore be reopened.

The multi read, single write system is adequate when an application keeps a file open for the duration that it is editing the file. However, consider the case of a program which loads all, or part, of a file, closes the file and then edits it. When the editing is complete, the file is rewritten to disc. A classic case of this is a word processor, which while not a database as such can be thought of in a similar way. One user might load in a text file, edit it over a period of, perhaps, hours, and then save it again. If somebody else using another station on the network had loaded the file after the original person, edited it and saved it back again, then all their changes would be lost when the original person saved his version.

The solution to this new problem is use a *lock bit*, which is a single status flag connected to the file in some way. Any program that is working on a file can set the lock bit for the file to signal to other programs that the file should be left alone because it is in a transient state. One way of implementing a lock bit would be to include an additional field in our File Description Record (FDR) - see previous articles. Any program accessing the file could check this field, and use it to determine if any other program was currently working on the file.

However, the Beeb already provides us with the ability to lock files by using the command *ACCESS. This is primarily intended to prevent files being accidentally erased, but we can also use if for our locking purposes. If we assume that all files will normally be unlocked, then a program can lock a file using *ACCESS while it is editing its contents, and then unlock it again afterwards. Should any other program try to access the file while it is locked, it will be able to load it, but will be prevented from saving it again. Incidentally, at this point it is worth noting that a locked file can still be modified using OPENUP. If you wanted to check the lock bit before opening the file, you could use the following function.

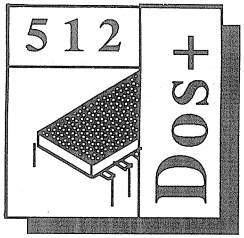
```
1000 DEF FNchecklock(filename$)
1010 DIM buff 17,name 20
1020 $name=filename$
1030 !buff=name
1040 A%=5
1050 X%=buff MOD &100
1060 Y%=buff DIV &100
1070 CALL &FFDD
1080 =(buff!14 AND 8)=TRUE
```

This routine should be called with the filename, for example:

```
locked=FNchecklock("DATAFILE")
```

and will return TRUE if the file is locked, and FALSE if it isn't.

We are nearing the end of this lengthy series on file handling. In the final part next month we will take a look at the practical aspects of writing file handling programs on a Beeb, and give some hints for debugging such programs. B



How to obtain cheap Software

This month Robin Burton looks at public domain and shareware software, a subject that may well interest more than just 512 owners.

A few of you have written to me about public domain and shareware software, and while some of you may be aware of these sources of DOS software, it occurred to me that others, especially the newer converts to the 512, may not be.

PUBLIC DOMAIN SOFTWARE

Public domain software has been around for quite some time (certainly it existed for CP/M machines before DOS became the standard business micro operating system) and is now common to many micros, though not with so wide a range of software as for DOS.

The concept is quite simple. Authors of software may or may not retain their copyright on the programs they have produced, but in either case they have decided to make them available free of charge to users. Depending on where you get it from, you may pay a small fee for the disc and copying of the software, but there is no fee for the programs, and you are at liberty to pass them on to other users.

Naturally, the quality of the software varies from one offering to another, but given the fact that it's free this can hardly be a complaint, and some of it is very good. If you visit your local library or enquire at your local computer bookshop, you will find that there are catalogues of public domain software, and for DOS systems there are literally hundreds of programs. Take a look, and you may be surprised at just what is available for virtually no cost.

SHAREWARE

Shareware is a sort of halfway house between public domain software and normally marketed commercial software. Like public domain, the idea originates in the USA.

Shareware is not free software, although you don't buy it when you first acquire it, and strictly speaking you don't buy it later either. This time the idea is that you can acquire a copy of the software for a nominal initial fee. This can vary from the cost of a disc plus postage to a few pounds. You then can try the programs on your own machine, using your own data so that you can judge its suitability for your needs.

If, having tried the software you like it, it does the job and you intend to continue using it, you are expected to send a registration fee, either direct to the author or to the local distributor. The registration fee is the author's reward for his efforts, and replaces what normally would be the purchase price if you were buying from commercial sources.

It goes without saying that this means the onus is on you to be honest and pay up if you do use the software. To fail to do so is of course theft, and moreover it's a perfect case of killing the goose that lays the golden egg, so I trust I need say no more on that point.

There are often inducements to help you to play fair too. Some of the larger shareware packages are supplied in a slightly 'cut-down' form. Although fully functional and working, on payment of the registration fee you will often be supplied with additional programs to enhance or extend the package, or perhaps a later version, these not being provided in the shareware copy.

Another point is that shareware programs are supplied on disc only, and while the disc will include quite adequate instructions (in text files) for using the programs, for the more complex packages registering your use will usually produce the benefit of proper documentation and manuals. Also many shareware authors actively support their work in the form of updates, enhancements, fixes for bugs, and some even provide direct phone or bulletin board support for individual user problems. Of course all of these are available only to registered users.

Finally, if all this wasn't enough, the registration fees are usually very much less than the software would cost if it was marketed conventionally. In spite of the lower prices and the somewhat unusual method of distribution, don't think that this implies that the software is in any way sub-standard, in fact quite the converse in many cases. Some shareware packages are really excellent, and many are as good

or better than the often over-priced, over-rated offerings from other suppliers. And frequently you can get commercially sold software as shareware at a fraction of the cost.

The reason for the lower prices is that the authors have no advertising or distribution costs and there's no software agency in the middle taking a cut either. In consequence part of the savings can be passed on to the user. The result is that you end up paying much less for a given package.

The case I quoted a few months ago of phoning the American distributors was a typical example. For that particular product an unusual situation exists, in that it is also commercially marketed in the U.K, but at three to four times the cost for the same software. That should give you an idea of the potential savings.

The benefit of this approach to software acquisition hardly needs pointing out, but to users of the 512 it's even more valuable. In our case the question of 'Will it run properly?' must be highest on the list. By means of the 'try before you buy' method, you don't have to part with large amounts of cash to find out if the software will run.

It's true that you can't return the disc for a refund if it doesn't work, but I've never seen a shareware disc that costs even as much as £10, less than most games for the BBC, so this is hardly a concern.

In addition, shareware has one thing in common with public domain software. You are not only at liberty, but are positively encouraged to pass on copies of the software to other users. The author will normally stipulate that you must pass on the entire contents of the disc just as you received it, with no amendments, additions or deletions, but apart from this there are no restrictions.

This means that if you are a member of a club, or have friends with 512s or PCs it is perfectly legitimate to share the initial fee and take a copy each (naturally sharing the cost does not apply to the registration fee).

I wondered about the value of including shareware as a Forum topic, but as some companies continue to exist by selling (at much higher cost) packages which are also currently available in shareware, it's obvious that a lot of people still don't know about this valuable, low cost, low risk method of acquiring software.

WHAT'S AVAILABLE?

The range of topics covered by shareware, even only in the U.K. and Europe, is really quite impressive. Most shareware catalogues will put the range of choices available at your local PC dealer to shame. The majority of topics don't offer just one or two packages either.

For example, in the catalogue of one distributor I've dealt with, Shareware Marketing, under each heading there are typically half a dozen or more different choices.In addition, a rating is given to each of the offerings, indicating their opinion of its quality, the range of its facilities, and whether it's recommended for first time, average or experienced users.

All the obvious applications are covered, including word-processors, spreadsheets, databases, graphics, CAD, project planning systems and so on. Language support is well catered for too, with plenty of library and utility routines for various versions of BASIC, 'C', PASCAL and even less common languages like FORTRAN.

There are also literally dozens and dozens of utilities, and as these tend to be smaller programs, a number are usually collected together to make up each disc, so one shareware utility disc might well contain a dozen or more different programs. These are usually grouped by type, so one disc will contain memory utilities, one keyboard programs, another disc utilities etc.

GUIDELINES

As with all other DOS software, 512 users have one overriding consideration which applies equally to shareware software, that is compatibility. All the usual problems can be found in shareware as frequently as elsewhere, but in some areas some fairly direct advice can be given.

In utility software, much more than in most full blown applications, execution speed and compact code is seen as a plus point. Because of this certain areas are much more likely to produce software which is incompatible with the 512 than others.

In general disc utilities will be trouble because, in order to gain speed and/or to provide clever facilities, the author will almost certainly have directly programmed the disc controller. Virtually all of these programs therefore fail in the 512. Unless 'legal' code is particularly advertised as a selling point, don't expect much success. In any case these

will cater only for standard PC formats, mainly 360K and 720K and so wouldn't be of very much value in the 512. Others, written for the quad-density 1.2 and 1.44Mb would be of no use whatever to us, even if they would run.

Keyboard utilities come next in the list of likely problem programs, including keyboard dependant utilities such as memory resident pop-ups. Some rely on 'legal' keyboard interrupts, and a few therefore do work in the 512. However, most expect to be able to detect the pressing of 'hot-keys (pre-defined special keypress combinations) at source. This they do by reading the keyboard hardware directly, so don't expect much success with them.

Memory utilities are probably next in line, mainly because many of them are also memory resident and use hot-keys. These which are run as separate transient programs on demand are mostly alright though, because they use the normal keyboard to application interface provided by DOS's CCP.

Finally, be wary of database packages or other disc dependant applications if they make a special point of stressing their high disc sorting or data handling speed. It's highly probable that these will use the same techniques as disc utilities to achieve their performance and will therefore produce the same problems in the 512.

Shareware is usually available in a variety of disc formats on both 3.5 and 5.25 inch discs, so remember to specify this if you take the plunge. Single density 360K is the safest choice. Because this is quite a small amount of data, disc space is at a premium and very little shareware is in a 'ready to run' form. Most is provided in a compressed form to reduce the physical space needed on disc, and it must be uncompressed before you can run it.

'PKXARC' is one of the programs frequently used to compress files, but there are others. Whatever method of compression was used, the program needed to uncompress your shareware will be included on the shareware disc, along with the instructions about how to use it. The space saved will vary with the type of file, but in extreme cases a compressed 360K disc may require almost twice as much when uncompressed.

All the compress/uncompress programs that I have used have proved to be legally written and I usually uncompress to an 800K 512 disc, so far always without any trouble. If for some reason you want to uncompress onto 360K discs make sure that you have two ready formatted for every source disc, though a better idea is to uncompress to 800K and then copy files to other disc sizes as needed.

You can read more about shareware software in Dabs Press 512 Master User Guide.

A catalogue of shareware products can be obtained from any of the following distributors:

SHAREWARE MARKETING, Beer, Devon EX12 3HW. Tel. (0297) 24088.

SELTEC, Northumberland House, Staines Business Centre, Staines, Middx TW18 2AP. Tel. (0784) 64257.

PC-SERVE, 1147 Greenford Rd, Greenford, Middx VB6 0DP. Tel. 01-864 2611.

Or you can log-on to one of the following bulletin boards:

Brown Bag Board 01-404 0897

Shareware Marketing (0297) 24090

PUBLICITY

Finally onto a different note, I'm pleased to say that someone has taken advantage of the offer made in the first 512 Forum. Richard Sterry has written from Wakefield asking for a mention of his informal 512 user group.

Richard is himself a member of the Wakefield BBC micro user group, a very enthusiastic and active group that holds meetings on the first Wednesday of every month, usually with a guest speaker giving a talk. Incidentally, the group also includes quite a few 512 users amongst its members, one of whom is the club's newsletter editor, Chris Hughes, who can be contacted on Wakefield 379778 if you'd like more information to join.

Richard's informal 512 user group, however, holds its 'meetings' in a rather different way and covers a rather larger area, using the U.K. Amateur 'Packet Radio' BBS network. This is a form of amateur radio which enables ASCII files to be sent between members in the U.K. and beyond, and operates much like a conventional bulletin board, except that there are no phone bills.

Richard says that radio amateurs active 'on packet' can join in simply by contacting him (call sign G4BLT) on his local BBS (GB7YAX). As the group is informal, there are no charges or fees.

Master 512 User Guide

Bernard Hill reviews a book which will be of interest to all 512 co-processor owners.

Title	**The Master 512 User Guide**
Publisher	**Dabs Press**
	5 Victoria Lane, Whitefield,
	Manchester M25 6AL.
	Tel. 061-766 8423
Price	**£9.95**

As most purchasers have found, buying a Master 512 co-processor is rather like driving a new car without a manual. You know roughly what it is supposed to do, but the details of exactly how it does it have to be found by exploration. As our 512 Forum contributor Robin Burton and his readers complain (see Beebug Vol.7 No.7) the Acorn manual is woefully inadequate, and a good replacement is long overdue. Once more the excellent Dabhand Guides fill the gap.

The Master 512 User Guide contains over 200 pages and at a price of £9.95 is excellent value for money in these days when (IBM) PC manuals and handbooks are often over £20. Actually it's part of a set of two: its companion volume, the Master 512 Technical guide (by Robin himself) will be available - says Dabs - at the end of June at £14.95.

The fact that it is the first of a pair actually highlights its principal difficulty: it walks a tightrope between being a replacement for the Acorn guide and being a full technical manual. In the main it performs this balancing act very well, but inevitably when reading it there are a number of times when I want to know more and indeed the book itself refers frequently to the Technical Guide ... just have to wait till June I suppose!

CONTENTS

The book consists of 19 chapters, a glossary, 7 appendices and an excellent index. The chapters can be grouped into an introduction (with a very clear coverage of directory structures and disc formats: rather brief on the PATH command); GEM (1 chapter); DOS+ commands (7 chapters of 90 pages); batch files (2 chapters); and a series of short chapters at the end on compatibility, expansion (now sadly out of date), shareware and a short question and answer chapter. Putting the Acorn guide to shame are the chapters on ED, PIP, and MEMDISK, which are covered at length. Sixty pages are devoted to a one-by-one listing of the DOS+ commands, both permanent and 'Transient' (i.e. supplied as programs on disc). Throughout, the explanations use helpful comparisons with other systems (such as ADFS, MS-DOS and CP/M), but little mention is made of the particular problems of the model B/B+ user - I suppose it is a Master 512 user guide.

The appendices cover software compatibility (a straight copy of the list available from Acorn); a DOS+ version compatibility chart and a supplied transient program list; GEM printer codes; Bulletin Boards and a Dabhand Guides Guide(!). There is also a one page appendix on the optional utilities disc.

This latter is available for £7.95 or £5 when ordered with the book. It contains five utilities: a reader for text files (enabling up and down scrolling); a dump utility (like *DUMP); a printer formatter, a customisable menu program, and disc sector editor, which can handle all 512 disc formats. Of particular interest is the inclusion of source code for these (written in C), but you would need a C compiler to modify them.

CONCLUSION

This book is suitable for beginner and experienced user alike: the layout is impeccable, the explanations clear and the index superb. Sections I would have liked to have seen expanded are those on the batch file processor (I really expected to see a description of how to set up your own menu system) and a 'Tips and Traps' chapter would not have been out of place. Particularly welcome, however is the section on shareware, which should be of interest to all 512 owners as a source of good software at reasonable cost. All in all highly recommended, and buy the disc too, if only for the disc sector editor. But if you want to know more, then you will need to read the 512 Forum in BEEBUG until the Technical Manual comes out. B

BEEBUG WORKSHOP

Spin a Disc (2)

David Spencer continues his look at disc systems.

In last month's Workshop we explained the layout of data on a floppy disc. This month we will look at the *disc controller* - the chip which interfaces the disc drive to the computer, and show how this can be controlled from your own software. Programming at this low level allows otherwise impossible functions to be implemented, such as a disc protection system.

There are two different disc controllers in common use on the Beeb, these being referred to by the part numbers of the controller chip. The first is the 8271. This was the original disc controller for the model B, but suffered from two problems. Firstly, the chip was almost obsolete when the Beeb was designed, and therefore was difficult to obtain, and very expensive. In fact, many companies sold second hand chips removed from old IBM machines. Secondly, the 8271 cannot support the MFM (double density) format as used by ADFS. The successor to the 8271 was the 1770. This more up-to-date controller, which supports both FM (DFS) and MFM (ADFS), is fitted to the Master, Compact, B+, Electron +3, and the newer model B upgrades. It is this chip we will concentrate on here.

ANATOMY OF A CHIP

At first sight you might think that the operation of the disc controller is relatively straightforward. All that is needed is a device which can move to the correct place on the disc, and then either read or write the data. However, when you consider in more detail what is involved, it all so seems much more complex.

For example, consider the reading of sector 4 on track 35 of a disc. The first step is to locate the correct track. If the disc controller knows which track the drive's head is over, it can calculate the necessary move. On the other hand, if it doesn't (as is the case when the drive is first turned on), the controller must first step the head out until the drive signals that it is over track 0, and then step back in to track 35. Next, the controller reads an ID field from the disc (see last month) to ensure that it is in fact on the correct track. Successive ID fields are then read until sector 4 is found, and then the data for that sector is read. Before transferring the data back to the computer, the controller must convert it from a serial format into a collection of eight-bit bytes. Finally, the CRC code is checked against the received data to detect any errors that may have occurred during the read operation.

PROGRAMMING THE 1770

To the programmer, the 1770 registers appear in four bytes of memory. The exact addresses depend on the computer, and these are listed in table 1. Also listed in the table is the address of a control latch which is external to the 1770. I will come back to this later.

Register	B/B+	Master/Compact
Status & Command	&FE84	&FE28
Track	&FE85	&FE29
Sector	&FE86	&FE2A
Data	&FE87	&FE2B
Control Latch	&FE80	&FE24

Table 1. Disc controller addresses

The 1770 is instructed to perform an operation (such as moving to a particular track or reading a sector) by writing a command to the command register. The full set of commands offered by the 1770 is fairly minimal, but adequate to perform all necessary operations. There is not room here to describe all the commands, though this information can be found in the 1770 data sheet which should be available from any supplier of the 1770. Table 2 gives a list of the more useful commands, along with the value in hex that is written to the command register to perform that particular command.

Command	Value	Hex
Restore	0 0 0 0 0 0 x y	&00-&03
Seek	0 0 0 1 0 0 x y	&10-&13
Step-in	0 1 0 1 0 0 x y	&50-&53
Read sector	1 0 0 0 0 0 0 0	&80
Write sector	1 0 1 0 0 0 P 0	&A0 or &A2
Write track (Format)	1 1 1 1 0 0 P 0	&F0 or &F2

Table 2. Reduced 1770 command set

Referring to table 2, you will see that there are six basic commands. The first of these is *Restore*, which returns the drive head to track zero regardless of where it is currently located, and sets the track register to zero. *Seek* steps the head until it is over the track whose number is given in the data register, while *Step In* moves the head one track towards the centre. *Read Sector* and *Write Sector*, not surprisingly, read and write a sector's-worth of data. Finally, *Write Track* writes an entire track including all the ID fields and gaps described last month. It is this command which is used to format a blank disc.

You will also notice from table 2 that the commands which cause the head to be moved between tracks have four different possible values. This is because the speed at which the head is stepped can be changed. There are four speeds, these being 6ms, 12ms, 20ms and 30ms between step pulses. These correspond to the four values for the commands. For example, command &01 moves the head to track zero at a rate of one pulse every 12ms, while command &13 will seek to a particular track at a stepping rate of 30ms. Most modern drives will work quite happily at the fastest rate, but some older drives will need 20 or 30ms pulses.

The commands that write data to the disc also have more than one value. This is to control a feature called *write precompensation* which will be explained next month.

THE STATUS REGISTER

Once a command has been started by writing a value to the command register, its progress can be monitored by reading the status register. Each of the eight bits in the status register has a different meaning, but for our purposes only five are relevant. These have the following meanings when set:

Bit 0 Command in progress
Bit 1 Data transfer requested
Bit 3 CRC error
Bit 4 Sector not found
Bit 5 Write protect error

Bit 0 is used to test if the command has completed or not. It will be set when the command is started, and reset once it has completed.

DATA TRANSFERS

Performing an operation such as *Restore* or *Seek* is fairly easy - all you have to do is issue the command (after setting the data register to the desired track for Seek) and then sit back checking the status register until the command is complete. However, in the case of data transfers, it is necessary to transfer data to or from the disc controller at a high rate (one byte every 32 micro-seconds in MFM mode). To achieve this, the disc controller generates a *non-maskable interrupt* (NMI) to the processor whenever a data byte needs to be transferred. This NMI causes the processor to suspend the current program almost immediately, and jump to a routine starting at address &D00 in RAM. This routine must then check that the interrupt is indeed a data request, and then transfer a single byte between the 1770's data register and RAM. All this must be done in less than 32 micro-seconds (about 20 instructions maximum allowing for the time to recognise the interrupt). To further complicate the situation, the 1770 also generates an interrupt when a command is

completed, or when an error occurs. Therefore, the interrupt routine must check for this as well.

THE CONTROL REGISTER

Before beginning to write a program that communicates with the disc controller directly, we need to look at the control register mentioned in table 1. Astute readers may have noticed that the 1770 commands provide no way of specifying which drive to use, which side of the drive to use, or whether to use FM (DFS) or MFM (ADFS). Instead, this information is stored in a separate *latch* chip called the *control register*, which is set by writing the appropriate value to the address given in table 1. The exact meaning of each bit depends on the computer, and is given in table 3. These will also be explained in more detail next month.

Bit 0	Drive zero select
Bit 1	Drive one select
Bit 2	Side select (model B and B+)
	1770 reset (Master)
Bit 3	FM/MFM select (model B and B+)
	Drive two select (Master)
Bit 4	Not used (model B)
	Interrupt enable (B+)
	Side select (Master)
Bit 5	Not used (model B)
	1770 reset (B+)
	FM/MFM select (Master)

Table 3. Control register functions

A GENERAL READ-WRITE ROUTINE

We now have all the knowledge needed to write a routine which will communicate directly with the 1770. To put the theory into practice we shall develop a generalised disc access routine which allows any number of sectors to be read or written to a disc, or any track to be formatted. Furthermore, our routine will be equally at home with both DFS and ADFS discs. For the sake of robustness and elegance, the routine will take the form of a ROM image, and will be called via an OSWORD call, although it could equally well be implemented in main memory.

Before getting to grips with the code, we need to define what parameters must be passed to the routine. Clearly, we need to be able to tell the routine which operation to perform (read, write or format), and whether to use FM (DFS) or MFM (ADFS) mode. We also need to specify where on disc the data is, and we will do this in terms of a drive number, a side number, a track number, and for read and write a sector number. Finally, we need to specify where in memory to transfer the data to or from, and for read and write the number of sectors to transfer. From this information we can devise a *parameter block* which will hold all the necessary information. This will consist of nine bytes of memory with the following meanings:

Byte 0	Command	
	(0=read, 1=write, 2=format)	
Bytes 1-4	Address of data in memory	
Byte 5	Bits 0-1	Drive number
	Bit 2	Side number
	Bit 3	FM(1) or MFM(0)
Byte 6	Track number	
Byte 7	Sector number	
Byte 8	Number of sectors	

We shall use OSWORD call number 115 for our disc command, because it doesn't clash with those provided by any other common ROMs. An OSWORD call is issued by loading the A register with the call number, and the X and Y registers with the low and high bytes of the address of the parameter block. So, for example, to write the contents of memory between locations &1200 and &12FF onto side 0, track 0, sector 1 of an ADFS disc in drive 1, you could use the following code:

```
?&900=1        : REM write command
!&901=&1200    : REM address
?&905=0        : REM drive etc.
?&906=0        : REM track number
?&907=1        : REM sector number
?&908=1        : REM just one sector
A%=115
X%=&900 MOD &100
Y%=&900 DIV &100
CALL &FFF1     : REM call OSWORD routine
```

We have covered a lot of technical information this month. In next month's workshop we will develop the code for our general purpose read/write routine, and give a full explanation of its functioning, which should clarify the details given here. Also, to illustrate the use of such a routine, we will use it to implement a high-speed ADFS formatter. B

MEWsoft's Fancy Labeller

If you ever need to print labels with as little fuss as possible, but with fancy borders and various typefaces at your disposal, then MEWsoft's Fancy Labeller could be the answer. David Somers samples the offering, and soon finds everything in sight labelled up!

Product	**Fancy Labeller**
Supplier	**MEWsoft,** **11 Cressey Road, London NW3 2NB.** **Tel. 01-267 2642**
Price	**£12.95 incl. VAT**

Readers of BEEBUG might be familiar with the products from MEWsoft: they are simple, but nevertheless useful programs, available to help with everyday life. The Fancy Labeller is no exception to this.

Producing labels, whether to grace Auntie's latest batch of homemade marmalade, or use as an attractive bookplate in the library collection, has never been an easy task. With MEWsoft's Fancy Labeller, these and others can be produced quickly and easily, in a matter of minutes. An Epson-compatible printer is required, capable of implementing the Plotter Graphics mode.

Designing a label

When the software is booted, the working environment is entered. This represents a menu selection at the top of the screen, with the work area below.

A label consists of three components: its size, the frame, and the text. From the main menu, the *Set Values* option allows the size of the label to be modified. You can also specify, amongst other things, how many labels are to be printed across the page and down the page.

Once the size has been correctly specified, a border needs to be retrieved from the disc. Five different types are supplied as standard. The name of a border is entered, which is then loaded and displayed. If you enter an invalid name, a list of the currently available frames is displayed for selection.

It is then necessary to place your text on the label. It is done by selecting the *Edit* option, moving the cursor to the desired place and typing away. Basic text manipulation is available to assist in the design, i.e. inserting or deleting of lines, and shifting the text on a line to the left or to the right.

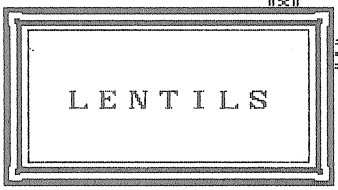
In addition to the computer's typeface, another three, supplied with the system, can be used on the labels: a modern face which resembles Century Schoolbook, a brush which resembles Flash Roman, and a sans serif which resembles Eurostyle. The required typeface is loaded by choosing the *Load Font* option and entering the name; a menu selection indicates those available if an invalid file is quoted.

When editing the text, F1 toggles between the computer's typeface and the loaded one. These alternative fonts are double size, i.e. each character occupies 2 by 2 characters of the computer's typeface. When the two typefaces are used carefully, the results can be most effective.

Although the fact that only one 'soft' font can be used at a time (in addition to the computer's one) might appear to be a limitation, it isn't. For most purposes, the use of more than two

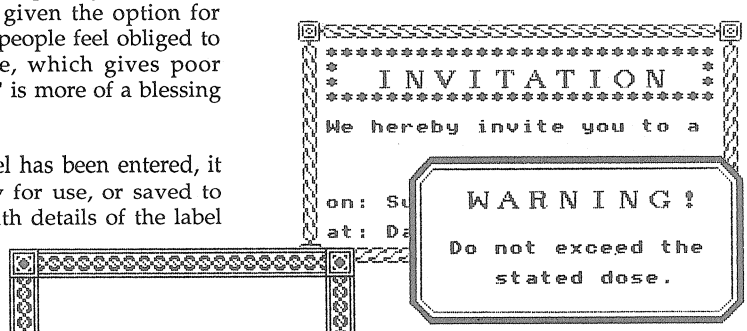
typefaces distracts from the aim, i.e. to communicate information quickly and in an aesthetic manner. When given the option for multiple typefaces, most people feel obliged to use as many as possible, which gives poor results; so this 'restriction' is more of a blessing in disguise.

Once the text for the label has been entered, it can be printed out ready for use, or saved to disc if required, along with details of the label size, typeface, etc. The only information which is not saved is the detail of how multiple labels



are to be printed; a minor irritant.

Throughout use, the Fancy Labeller is extremely easy and intuitive to work with. Indeed, little reference was made to the instruction sheets, which are only needed when configuring the system for multiple column labels, where helpful suggestions are made.



Again, MEWsoft have produced a simple, but effective piece of software that could be put to a multitude of tasks, and should find itself easily used by one and all.

Additional fonts and borders are available for just £6.95 (for approximately 8 fonts and 18 borders). A font and border designer kit is also available, for £8.95, which enables existing ones to be edited, or new ones designed. These will be examined in a future issue of BEEBUG. B

First Course - Investigating Teletext Mode (continued from page 37)

You will then see the time (maximum 3 figures) displayed in large cyan digits (code 150) on the screen. It is then that the inclusion of 'space' as a valid character shows its worth, by wiping out unwanted digits as the number reduces. You may like to try and improve the display by devising your own designs for the digits 0 to 9.

THE FORMULA EXPLAINED

We will now examine how the formula in line 1060 was obtained. Figure 3 shows how a four digit display would be laid out on the screen (remember that the 40 columns in mode 7 are numbered from 0 to 39). Four digits will need 24 columns (4*6), leaving 16 (40-4*6) over. Division by 2 gives the number of spaces before and after the digits, and because counting starts at zero, this also gives the starting position:

```
x=(40-4*6)/2
```

If we perform the same calculation when the number of digits is specified as 'd' we get:

```
x=(40-6*d)/2
```

which can be simplified to:

```
x=20-3*d
```

To get the starting position for the second digit we need to add 6 to this, 2*6 for the third digit and so on. In other words 6 multiplied by one less than the position of the digit in the number. In the procedure, I% counts the digit position (from right to left, but that doesn't matter), so the starting position of the digit in position I% is given by:

```
x=20-3*d + 6*(I%-1)
```

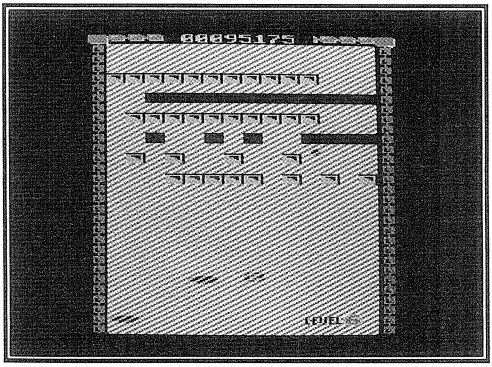
and there we have it. Rough diagrams will often help in these situations, but otherwise just clear careful thinking is what is required.

The magazine disc/tape contains four example programs to demonstrate the techniques described in this article. B

Bouncer Ball

Matt Eastmond, author of previous BEEBUG games, presents his latest - a much enhanced version of a well loved classic.

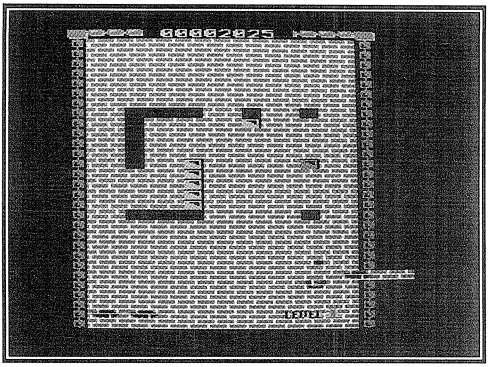
This is an updated version of Breakout, a highly acclaimed game from the earlier days of micros. The scenario is as follows. Having been reduced to the size of a proton and sucked into a parallel universe, you materialise as a bat, with a mission to reach level fifteen by knocking out all the bricks.



Use Z and X for left and right, Return to fire. Bounce the ball and destroy all the bricks to move to the next level. Spin the ball by moving the bat whilst striking it. Some of the bricks, on destruction, drop a bonus capsule. Collect this and one of four benefits will automatically ensue:

1. A portal will open to allow free passage to next level.
2. The ball will become stuck to the bat - to release it press fire (Return).
3. You are awarded an extra bat.
4. A letter of the word bonus will be lit up. If you manage to light the whole word you receive a massive bonus score.

The program consists of two listings; type these in and save them as BOUNCE1 and BOUNCE2. To play the game, simply run the first program. This defines all the characters and sound envelopes (and assembles a very short machine code program for single key input) before chaining in turn the second (main) program. No one has yet completed the game without cheating.



The magazine disc/tape contains an extended version of the game which allows you to design your own screens, as well as use those built into the program.

Listing 1

```
   10 REM Program Bounca Ball
   20 REM Version B1.0
   30 REM Author Matt Eastmond
   40 REM BEEBUG May 1989
   50 REM Program subject to copyright
   60 :
  100 MODE7:VDU23,1;0;0;0;0;
  110 ENVELOPE1,135,16,12,20,2,1,1,127,0
,0,-55,126,0
  120 ENVELOPE2,1,4,-4,4,10,20,10,127,0,
0,-5,126,126
  130 ENVELOPE3,1,0,0,0,1,1,1,60,-10,-10
,-10,126,100
  140 ENVELOPE5,2,0,48,-48,5,1,1,110,-1,
-2,-3,126,0
  150 ENVELOPE6,1,1,0,0,4,60,60,80,70,0,
-1,80,110
  160 ENVELOPE10,1,0,0,0,1,1,1,60,-4,-1,
-1,126,100
  170 VDU23,224,51,102,204,153,51,102,20
4,153
  180 VDU23,225,8,62,62,46,46,38,62,8
  190 VDU23,226,0,126,70,223,126,126,0,0
  200 VDU23,227,0,255,255,255,255,255,25
5,255
  210 VDU23,228,0,0,15,63,127,127,31,5
  220 VDU23,229,0,0,224,248,252,248,240,
160
```

```
  230 VDU23,230,96,240,240,240,96,0,0,0
  240 VDU23,231,0,254,194,242,250,254,25
4,254
  250 VDU23,232,20,34,65,128,1,130,68,40
  260 VDU23,233,0,0,77,73,77,73,73,108
  270 VDU23,234,0,0,90,82,90,82,82,155
  280 VDU23,235,17,40,68,130,17,40,68,13
0
  290 VDU23,236,247,247,247,0,191,191,19
1,0
  300 VDU23,237,0,0,255,255,255,255,255,
255
  310 VDU23,238,4,64,224,68,14,132,32,0
  320 VDU23,239,64,225,72,28,8,33,112,33
  330 VDU23,240,32,114,32,128,9,67,225,6
4
  340 VDU23,241,0,16,57,16,0,130,7,34
  350 VDU23,242,255,129,129,129,129,129,
129,255
  360 VDU23,243,128,128,128,129,65,62,12
8,128
  370 VDU23,244,126,195,219,199,219,195,
126,0
  380 VDU23,245,0,127,179,213,234,85,63,
0
  390 VDU23,246,0,254,171,85,171,86,252,
0
  400 VDU23,247,188,198,206,248,204,198,
188,0
  410 VDU23,248,120,204,134,198,206,252,
120,0
  420 VDU23,249,32,112,230,198,206,254,1
20,0
  430 VDU23,250,196,230,246,222,206,198,
198,0
  440 VDU23,251,60,126,98,192,192,252,12
4,0
  450 VDU23,252,12,28,50,126,102,230,198
,0
  460 VDU23,253,96,96,192,192,192,248,24
8,0
  470 VDU23,254,60,126,96,124,14,14,124,
0
  480 P%=&80:[OPT2:LDA#135:JSR&FFF4:STX&
70:RTS:]
  490 T%=INKEY-256
  500 IF T%=253 OR T%=245 OR T%=244 THEN
 ?&71=231:?&72=224 ELSE ?&71=135:?&72=12
8
  510 CHAIN"BOUNCE2"
```

Listing 2

```
   10 REM Program Bounca Ball Game (2)
   20 REM Version B1.1
   30 REM Author Matt Eastmond
   40 REM BEEBUG May 1989
   50 REM Program subject to copyright
   60 :
  100 IF PAGE<=&E00 GOTO150
  110 VDU21:*TAPE
  120 *KEY0FORA%=0TO(TOP-PAGE)STEP4:A%!&
E00=A%!PAGE:NEXT|MPAGE=&E00|MOLD|MVDU6:G
OTO10|M
  130 *FX138,0,128
  140 :
  150 DIM N$(6),S%(6)
  160 ON ERROR GOTO 310
  170 RESTORE2710:FOR T=0 TO 6
  180 S%(T)=5000*(T+1):READ N$(T):NEXT
  190 MODE2:VDU23;10,32;0;0;0;
  200 PROCkeys:REPEAT
  210 MODE2:VDU23;10,32;0;0;0;
  220 PROCsetup:REPEAT
  230 PROCcanon:PROCball:IF M% PROCfall
  240 IF Y%<32 MOVE Z%,Y%:VDU230:L%=0:Y%
=240:F%=16:liv%=liv%-1:MOVE X%,224:VDU22
8,229:PROClives:PROCwow:IF liv%>-1 PROCt
ogether:MOVE Z%,Y%:VDU230
  250 IF t%=tt% OR X%=1120IF e%=0MOVE Z%
,Y%:VDU230:scr%=scr%+1:s%=s%+5000+scr%*1
000:PROCscor:VDU31,17,25,17,6,225,32,32,
28,0,12,1,2,12,26:L%=0:P%=0:Y%=240:F%=16
:M%=0:T%=0:S%=137:O%=0:PROCscreen:t%=1
  260 UNTIL liv%=-1 OR t%=tt% AND e%=1
  270 IF s%>S%(6) PROCenter
  280 UNTIL FALSE
  290 END
  300 :
  310 MODE7:REPORT:PRINT" at line ";ERL
  320 END
  330 :
 1000 DEF PROCcanon
 1010 A%=X%
 1020 IF INKEY-67 IF X%<976 X%=X%+48
 1030 IF INKEY-98 IF X%>208 X%=X%-48
 1040 IF INKEY-74IF L%=2L%=1:E%=0:F%=32
 1050 IF O%=1 IF X%>975 IF L%<2 IF INKEY
-67 X%=X%+48
 1060 IF A%=X% ENDPROC
 1070 MOVE A%,224:VDU228,229
 1080 MOVE X%,224:VDU228,229
 1090 ENDPROC
 1100 :
 1110 DEF PROCfall
 1120 MOVE M%,N%:VDU244
 1130 N%=N%-32
 1140 MOVE M%,N%:VDU244
 1150 IF N%<32 MOVE M%,N%:VDU244:M%=0
 1160 IF X%-M%>-96 IF X%-M%<66 IF N%-224
<32 IF N%-224>-32 PROCbonus
 1170 ENDPROC
 1180 :
 1190 DEF PROCbonus
 1200 SOUND1,3,149,1:s%=s%+750:PROCscor
 1210 MOVE M%,N%:VDU5,244
```

```
 1220 M%=0:L%=0
 1230 IF scr%=15 ENDPROC
 1240 IF RND(scr%+1)=1 IF e%=0IF scr%<14
 VDU4,31,17,25,17,14,17,143,BK,BK,BK,17,
128,5:O%=1:ENDPROC
 1250 IF scr%<14IF RND(4)=1L%=1:ENDPROC
 1260 IF RND(9)=2 IF scr%<10IF liv%<5 SO
UND1,1,140,8:liv%=liv%+1:PROClives:PROCt
ime(40):ENDPROC
 1270 RESTORE2720:FOR Y=0TO P%:READ T:NE
XT:PROCblur(CHR$(T),64,(768-(P%*32))):P%
=P%+1:GCOL3,6:IF P%=5 SC%=SC%+50000:PROC
scor:FORT=1TO6:VDU4,31,0,6+T,32,32,5:SOU
ND1,10,200,2:PROCtime(20):NEXT:P%=0
 1280 ENDPROC
 1290 :
 1300 DEF PROCball
 1310 T%=T%+1:IF T%=5 T%=0:S%=137
 1320 B%=Z%:C%=Y%
 1330 IF L%=2 E%=0:F%=0:Z%=X%+40:Y%=240
 1340 Z%=Z%+E%:Y%=Y%+F%
 1350 IF Z%<2000R Z%>1072 E%=-E%:SOUND3,
3,0,1:b%=0
 1360 IF Y%>952 F%=-F%:b%=0:SOUND2,5,157
,1:IF F%<-24 F%=F%-8
 1370 IF Z%-X%<128 IF Z%-X%>-1 IF Y%<240
IF Y%>160IF b%=0PROChit
 1380 MOVE B%,C%
 1390 VDU230,4,31,Z%DIV64,32-Y%DIV32
 1400 CALL&80:IF ?&70=?&71 VDU17,14,17,1
43,BK,17,0,17,128:SOUND1,1,S%,3:s%=s%+75
*scr%:F%=-F%:PROCscor:b%=0:t%=t%+1:S%=S%
+5:T%=0:IF M%=0IF RND(5)=1SOUND2,2,100,1
:M%=B%:N%=C%:MOVE M%,N%:VDU5,244:MOVE Z%
,Y%:VDU230:ENDPROC
 1410 MOVE Z%,Y%:VDU5,230
 1420 ENDPROC
 1430 :
 1440 DEF PROClives
 1450 VDU4,31,3,29,17,14,17,143,BK,BK,BK
,BK,BK,BK,BK,BK,5,17,128
 1460 IF liv%<1 ENDPROC
 1470 FOR T=192TO (liv%*128)+64 STEP128
 1480 MOVE T,96:VDU228,229
 1490 NEXT
 1500 ENDPROC
 1510 :
 1520 DEF PROChit
 1530 b%=1:F%=RND(24)+8:E%=RND(40)-20
 1540 SOUND2,10,53+scr%*5,9
 1550 IF INKEY-67 IF RND(2)=1 F%=RND(32)
:E%=-RND(64):Y%=256
 1560 IF INKEY-98 IF RND(2)=1 F%=RND(32)
+8:E%=RND(64):Y%=256
 1570 IF F%<8 F%=8
 1580 IF L%=1 L%=2
 1590 IF F%<32 IF scr%>9 F%=32
 1600 ENDPROC
 1610 :
 1620 DEF PROCscor
 1630 IF s%>99999999 ENDPROC
 1640 VDU4:?&34F=24:?&30A=25:COLOUR1
 1650 PRINTTAB(6,1);" ";STRING$(8-LEN(ST
R$(s%)),"0");s%;" "
 1660 ?&34F=32:?&30A=19
 1670 ENDPROC
 1680 :
 1690 DEF PROCscreen
 1700 IF scr%=16 VDU4,28,3,30,16,2,12,26
:?&34F=24:?&30A=25:COLOUR1:PRINTTAB(7,9)
;"Well done":PRINTTAB(5,11);"Now try aga
in!":SC%=SC%+100000:PROCscor:PROCtime(20
0):scr%=1
 1710 RESTORE 2730:IF scr%>1 SOUND1,2,10
1,1:FORT=1TO scr%-1:READ a%,a%,a%,a%,A$:
NEXT
 1720 VDU26:READ tt%,b1,b2,BK,A$
 1730 VDU19,14,0;0;19,15,0;0;17,14
 1740 VDU28,3,30,16,2,12,26,17,143
 1750 BK=BK+224:FORA%=2TO30
 1760 PRINTTAB(3,A%)STRING$(14,CHR$BK)
 1770 NEXT
 1780 VDU19,14,b1;0;19,15,b2;0;17,128
 1790 FORT=1TO LEN(A$) STEP5
 1800 X=ASC(MID$(A$,T,1))-47:Y=ASC(MID$(
A$,T+1,1))-47:L=ASC(MID$(A$,T+2,1))-47:D
%=ASC(MID$(A$,T+3,1))-47:C=ASC(MID$(A$,T
+4,1))-47:COLOURC
 1810 FOR A%=0TO L
 1820 IF D%=1 VDU31,X+A%,Y,231
 1830 IF D%=0 VDU31,X,Y+A%,231
 1840 IF D%=2 VDU31,X+A%,Y+A%,231
 1850 NEXT:NEXT
 1860 VDU5:RESTORE2720:GCOL0,2
 1870 FOR Y=0TO4:READ T
 1880 MOVE 64,(768-(Y*32)):VDU T:NEXT
 1890 GCOL0,1:MOVE 840,92:VDU233,234
 1900 GCOL0,2:MOVE 832,96:VDU233,234
 1910 PROCblur(STR$(scr%),968,92)
 1920 GCOL3,6:PROClives
 1930 PROCtogether:MOVE Z%,Y%:VDU230
 1940 ENDPROC
 1950 :
 1960 DEF PROCwow
 1970 SOUND1,3,99,0
 1980 SOUND0,6,3,1:FORT=0TO3
 1990 MOVE X%,220:VDU238,239
 2000 PROCtime(4)
 2010 MOVE X%,220:VDU240,241
 2020 PROCtime(4):NEXT
 2030 ENDPROC
 2040 :
 2050 DEF PROCtogether
 2060 Z%=960:MOVE 160,224:VDU228:MOVE 99
2,224:VDU229:FORX%=192 TO 544 STEP32:MOV
E X%-32,224:VDU228:MOVE X%,224:VDU228:MO
```

```
VE Z%+32,224:VDU229:MOVE Z%,224:VDU229:Z
%=Z%-32:NEXT:X%=X%-32
 2070 ENDPROC
 2080 :
 2090 DEF PROCmo2
 2100 FORT=1TO7:VDU19,T,0;0;5:NEXT
 2110 PROCtitle
 2120 REPEAT:A=0:T=0:REPEAT
 2130 IF S%(T)<S%(T+1) D=S%(T):D$=N$(T):
S%(T)=S%(T+1):S%(T+1)=D:N$(T)=N$(T+1):N$
(T+1)=D$:A=1
 2140 T=T+1:UNTIL T=6:UNTIL A=0
 2150 Y=0:FORT%=8TO 21STEP2:COLOUR Y+1
 2160 PRINTTAB(3,T%);Y+1;" ";N$(Y)
 2170 PRINT TAB(10,T%);" ";STRING$(8-LEN
(STR$(S%(Y))),"0");S%(Y)
 2180 Y=Y+1:NEXT
 2190 FORT=1TO7:VDU19,T,T;0;:NEXT
 2200 REPEAT UNTIL INKEY-99 OR INKEY-35
 2210 ENDPROC
 2220 :
 2230 DEF PROCtitle
 2240 PROCblur(CHR$247+CHR$248+CHR$249+C
HR$250+CHR$251+CHR$252+CHR$32+CHR$247+CH
R$252+CHR$253+CHR$253,312,920)
 2250 ?&34F=24:?&30A=25:VDU4:COLOUR1
 2260 PRINTTAB(4,5)"Matthew Eastmond "
 2270 PRINTTAB(6,24)"Press space"
 2280 ENDPROC
 2290 :
 2300 DEF PROCsetup
 2310 liv%=2:scr%=1:s%=0
 2320 VDU19,9,2;0;19,8,2;0;:PROCmo2
 2330 VDU12,19,6,0;0;19,4,0;0;
 2340 FORY%=2TO30
 2350 VDU17,6,31,2,Y%,225,31,17,Y%,225
 2360 NEXT:VDU5
 2370 A$=CHR$227+STRING$(14,CHR$226)+CHR
$227:MOVE 136,988:GCOL0,4:PRINT;A$
 2380 MOVE 128,992:GCOL0,6:PRINT;A$
 2390 PROCscor:VDU19,6,6;0;19,4,4;0;
 2400 e%=0:Y%=512:E%=3:F%=32:Z%=640
 2410 PROCscreen
 2420 b%=0:t%=1:X%=544:L%=0:T%=0:S%=137
 2430 M%=0:N%=0:O%=0:P%=0
 2440 ENDPROC
 2450 :
 2460 DEF PROCkeys
 2470 VDU5:PROCtitle:COLOUR2
 2480 PRINTTAB(5,11)"Z ..... LEFT";TAB(
5,13);"X ..... RIGHT";TAB(5,15);"RETURN
. FIRE"
 2490 REPEAT UNTIL INKEY-99:CLS
 2500 SOUND1,5,101,3
 2510 ENDPROC
 2520 :
 2530 DEF PROCblur(A$,x,y)
 2540 GCOL0,4:MOVE x-8,y-4:PRINT A$
 2550 MOVE x-8,y+4:PRINT A$
 2560 MOVE x+8,y+4:PRINT A$
 2570 MOVE x+8,y-4:PRINT A$
 2580 GCOL0,6:MOVE x,y:PRINT A$
 2590 ENDPROC
 2600 :
 2610 DEF PROCenter
 2620 FORY%=1TO31:VDU4,31,0,0,11
 2630 PROCtime(0):NEXT
 2640 ?&34F=24:?&30A=25:COLOUR1
 2650 PRINTTAB(4,7)"Enter your name?"
 2660 VDU31,7,9,17,2,28,7,10,16,10
 2670 *FX15 0
 2680 INPUT N$(6):S%(6)=s%:VDU26:IF LEN(
N$(6))>7 A$=N$(6):N$(6)=LEFT$(A$,7)
 2690 ENDPROC
 2700 :
 2710 DATA T.o.G.,John,Owen,Matt,Jaybub,
Beebug,Da Boss
 2720 DATA 247,248,250,249,254
 2730 DATA 41,0,1,12,488014B801493/47=3/
3:93/2==3/5
 2740 DATA 43,4,0,8,4:5/3=:5/16;4026=401
6<40449804
 2750 DATA 27,5,0,0,463126631486310:6315
48111<6112
 2760 DATA 31,0,4,12,2880427600264012520
524002
 2770 DATA 31,5,0,11,8510076301675036850
4793008:102
 2780 DATA 69,0,0,0,2490046:01289034::04
2<9025>900
 2790 DATA 49,4,0,14,34800356/145602464/
456403572/567200680/1
 2800 DATA 36,4,0,8,2=7112;7102971427615
 2810 DATA 43,1,0,19,3=:024<8035;6016:40
57920688001
 2820 DATA 46,0,5,12,567/0767/086100:76/
08>0002@<01
 2830 DATA 47,1,0,14,466/1575/2693/47:2/
1::2/3;93/2<84/1=75/4
 2840 DATA 47,0,0,8,29300625/02=3006>4/0
;25/0;9300;=300;>4/0
 2850 DATA 53,0,4,0,38:00596013::005;601
3<:00
 2860 DATA 26,0,4,20,558/2755/08;100994/
0
 2870 DATA 127,4,0,2,24<1026<1128:142:81
52<6102>4122@2102B01522<1542:12628148261
2:2410<2211>2015
 2880 DATA 127,4,0,2,24<1026<1128:142:81
52<6102>4122@2102B01522<1542:12628148261
2:2410<2211>2015
 2890 :
 2900 DEF PROCtime(t)
 2910 TIME=0:REPEAT:UNTIL TIME>t
 2920 ENDPROC
```

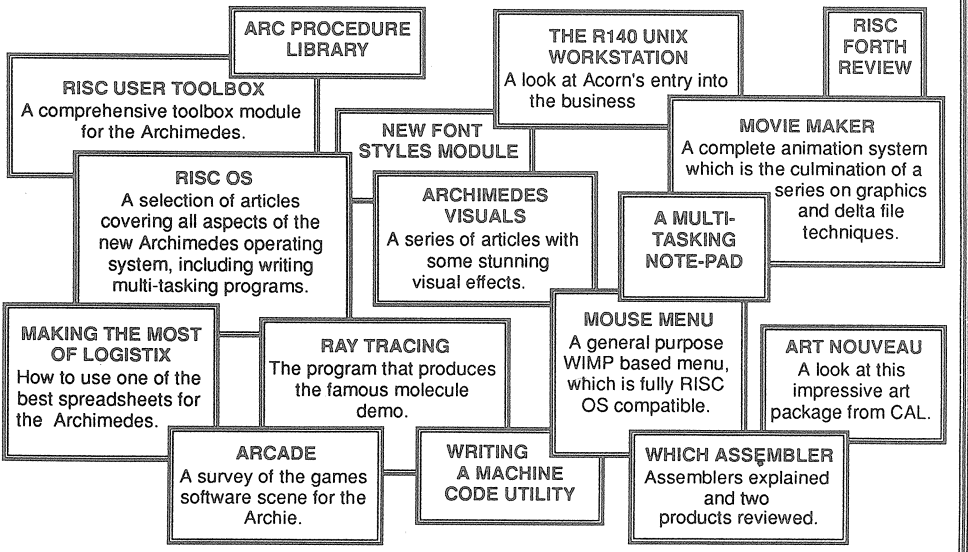
RISC USER

The Archimedes Support Group

Our Risc User magazine is now in its second volume and is enjoying the largest circulation of any magazine devoted to the Archimedes. The list of members seeking support from the Risc User group is growing steadily and as well as private individuals includes schools, colleges, universities and industry and government establishments.

Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines. A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer, particularly at this time while documentation on the Archimedes and RISC OS is still limited.

Here are just some of the topics covered in more recent issues of RISC User:



Don't delay - Phone your instructions now on (0727) 40303

As a member of BEEBUG you may extend your subscription to include RISC User for only £8.50 (overseas see below).

Name:.....
Memb No:.....
Address: .....
.....
.....
.....

SUBSCRIPTION DETAILS	
Destination	**Additional Cost**
UK,BFPO &Ch Is	£ 8.50
Rest of Europe and Eire	£13.00
Middle East	£15.00
Americas and Africa	£17.00
Elsewhere	£19.00

I wish to receive both BEEBUG and RISC User. *I enclose a cheque for £ ..... or alternatively*

I authorise you to debit myACCESS/Visa/Connect account: _____/ _____/ _____/ _____/

Signed: ..... *Expiry Date:* _____/ _____/

Send to: RISC User, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX, or telephone (0727) 40303

UK Bulletin Boards

All the listed boards were believed to be active as of April 1989. Circumstances may have changed since then and BEEBUG would appreciate members letting us know of any changes they have discovered, such as boards that have ceased operating or changed times and baud rates. Also, if you know of a board that is not listed here, please let us know for the benefit of other members.

Some new boards, particularly NBBS and OBBS systems, enable Beeb owners with Commstar or the Demon Zromm to have colour. With Commstar use mode 7 and switch off the filter mask at the command screen before entering 'Chat' mode. Zromm users should use Mode 7 and *Chat instead of *Terminal. Then when logged on answer 'Yes' to the question 'Are you using software on a BBC that allows colour'.

LONDON

Arrakeen
24 hrs
(01) 738 7304
V21,V23

Body Matters
24 hrs
(01) 603 7581
V21,V23,V22,V22bis
medical topics

Brixton ITeC
24 hrs
(01) 735 6153
V23

BrownBag
24 hrs
(01) 591 6687
V21,V22,V22bis

CIX
24 hrs
(01) 399 5252
V21,V23,V22,V22bis
multi user, conferencing, subscription

Communitel
24 hrs
(01) 988 7402
V23

Co-op Board
24 hrs
(01) 316 8488
V21,V23,V22,V22bis
PC & BBC

The Crown Green
24 hrs
(01) 245 1512
V21,V23,V22,V22bis
run by the Post Office

Crystal Tower
24 hrs
(01) 888 2813
V21,V23,V22,V22bis
home of Programmer's Routine Library

Dataflex BBS
24 hrs
(01) 543 7020
V21,V22

DirectConnection
24 hrs
(01) 853 3955
V21,V23,V22,V22bis
multi user, subscription

Direct Line
24 hrs
(01) 841 1847
V21,V22,V22bis
PC support

Distel
24 hrs
(01) 679 6183
V21,V23
commercial

The Embasey
24 hrs
(01) 366 1778
V21,V23,V22,V22bis

The Emerald Tower
9am-5pm
(01) 405 8963
V21,V23

Finger Fone
24 hrs
(01) 490 1545
V21,V23,V22,V22bis
24 hrs
(01) 253 3065
V21,V22,V22bis

Gnome at Home
24 hrs
(01) 888 8894
V23
multi user, public areas and subscription

Hackney BBS
24 hrs
(01) 985 3322
V23

Health Data
24 hrs
(01) 986 4360
V23
medical topics

Heaven
24 hrs
(01) 994 9119
V21,V23
multi user games

Jolly Roger
24 hrs
(01) 742 1640
V21,V23

Kyberneels
24 hrs
(01) 873 7294
V21,V23,V22,V22bis
for church and charity computer users

London City Magazine 24 hrs	(01) 488 7848 V21,V23
London Connexion 24 hrs	(01) 657 3240 V21,V23,V22,V22bis subscription
London Mail Centre 24 hrs	(01) 543 1200 V21,V23,V22,V22bis
London Metropolis 24 hrs	(01) 519 1055 V21,V23,V22,V22bis MNP error correction
London U'gnd 24 hrs	(01) 883 0198 V21,V23,V22,V22bis multi user
Mac Tel Metro 24 hrs	(01) 543 8017 V21,V23,V22,V22bis multi user
Marctel 24 hrs	(01) 346 7150 V21,V23 FBBS system
MBBS Mitcham 24 hrs	(01) 648 0018 V21,V23
NNBBS London 24 hrs	(01) 455 6607 V21,V23
Nottingdale Tec Ctr 24 hrs	(01) 968 6033 V23
Organic Garden 24 hrs	(01) 464 3305 V21,V23,V22,V22bis gardening
OSI Lives 24 hrs	(01) 429 3047 V21 (ringback)
PC Access 24 hrs	(01) 606 0081 V21,V23,V22,V22bis MNP error correction, PC User Mag BBS, subscription
PC Server 24 hrs	(01) 864 2633 V21,V23,V22,V22bis MNP error correction
The Pink Triangle 10pm-8am	(01) 961 5808 V21,V22,V22bis
Prometheus 24 hrs	(01) 300 7177 V23 Astronomers' SIG
Pyramid 24 hrs	(01) 239 0871 V21,V23,V22,V22bis Programming and Mean 18 Golf
Sirius/WBBS 24 hrs	(01) 542 3772 V21,V23
Strange Brew 9pm-11pm,	(01) 688 5757 V22,V22bis Mean 18 golf courses
SW10 Warehouse 24 hrs	(01) 351 7262 V21,V23,V22,V22bis HST 9600 baud
TBBS Rovoreed 24 hrs	(01) 542 9767 V21,V23,V22,V22bis
Techno Line 24 hrs	(01) 450 9764 V23 commercial
Techno Line 2 MF, Evenings, WE, 24 hrs	(01) 452 1500 V23 commercial
Third Wave Systems 24 hrs	(01) 585 3163 V21,V23
Wonderland 24 hrs	(01) 880 5330 V23 MUG
Woodgreen BBS 24 hrs	(01) 889 5824 V21,V23

To be continued

BBC Acorn User Show

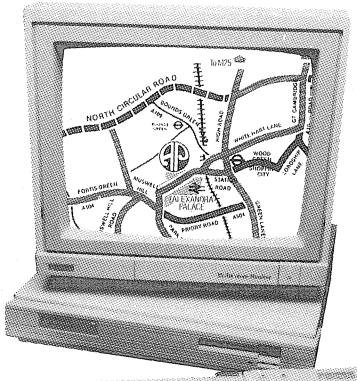
BBC ACORN USER, with the backing of the BBC and the enthusiastic support of Acorn Computers, have together planned what could be the most exciting computing event of the year.

The new, all-action BBC ACORN USER SHOW is back, at the first home of BBC TV – Alexandra Palace featuring:-

- new product launches
- new software
- massive computing exhibition
- informative seminars
- technical clinics
- workshops
- demonstrations

All this and more will make the new BBC ACORN USER SHOW a real must for everyone interested in computers and their applications.

There are daily competitions, free draws with fabulous prizes to be won and celebrity guest appearances.



THE 1989 BBC ACORN USER show

Facilities at the tastefully restored Alexandra Palace are simply superb and it's so easy to get to by road, rail or tube. If you're driving you'll be glad to know there are 2000 free car parking spaces. So why not make a day of it? Apart from the beautiful grounds you'll find something interesting and exciting around every corner.

Why not save time and money by booking your ticket in advance? Cut the coupon now and ensure your priority booking.

ALEXANDRA PALACE · LONDON
Friday July 21st 3pm-9pm
Saturday July 22nd 10am-6pm
Sunday July 23rd 10am-6pm

BOOK NOW AND SAVE

Advance Tickets: Adults £2.50 Under 16s £1.50

At the door: Adults £3.50 Under 16s £2.50

Please send me _____ Adult Tickets at £2.50 each

Please send me _____ Under 16s Tickets at £1.50 each.

I enclose cheque/PO for £_____ payable to SAFESELL Ltd., Market House, Cross Road, Tadworth, Surrey KT20 5SR.

DON'T SEND CASH.

Name _____

Address _____

RU1

The hints and tips for this issue have been compiled by Mike Williams. Remember that we pay £5 for each hint published and £15 for the star hint.

***** STAR HINT *****

GET AND GET AGAIN

David Spencer

The tilde '~' preceding any value (or variable) will ensure that on the screen (or printed out) the relevant value appears in hexadecimal format. In addition the tilde can be used with functions which return a numeric value to achieve the same result, and this includes Basic's built-in functions.

One that you may not consider is GET. Thus:

```
PRINT ~GET
```

will display the ASCII value obtained in hex. More unexpectedly:

```
PRINT ~GET GET
```

will display both bytes in hexadecimal, and this can be extended. In practice, a tilde will apply to all following values in a PRINT statement provided these are separated only by spaces, not commas or semi-colons. Thus:

```
PRINT ~A B C
```

will display the values of all three values in hexadecimal. Otherwise, a separate tilde will be needed with each GET. For example:

```
PRINT "&"+STR$~(GET);STR$~(GET)
```

would display:

```
&7F7F
```

in response to pressing Delete twice.

COUNTING THE DAYS

Paul R.Holmes

Programs often need to determine the number of days in a given month. This can be done by reading a series of numbers into an array, but the following formula is more convenient and elegant:

```
D%=30-((M%-(M%>7))MOD2=1)+2*(M%=2)
```

where D% is the number of days in month M%.

This works for all months except February in a leap year. The validity of the formula can be extended to all months from March 1900 to January 2100 by the following:

```
D%=30-((M%-(M%>7))MOD2=1)+2*(M%=2)-
   ((M%=2)AND(Y%MOD4=0))
```

where Y% is the year (all four or just the last two digits will suffice).

IF....

John Lasruk

The construction:

```
IF ... AND ... AND ... THEN ...
```

can in many cases be replaced by a series of 'IF's, and this will generally run much faster. The reason is that when Basic encounters AND it reads all the component parts before deciding what to do. In an IF sequence, Basic will stop at the first false IF it encounters.

In a single line the gain is negligible, but in a loop, it is nearly possible to halve execution time. The following program illustrates the point:

```
10W=1:X=2:Y=3:Z=4:T=0:@%=T
20PRINT" IF SERIES: ": NOW=TIME
30FOR A=1 TO 1000:IF W=1 IF X=9 IF
Y=3 IF Z=4 T=W
40NEXT:PRINT(TIME-NOW)/100;" seconds"
50T=0:PRINT"AND SERIES: ": NOW=TIME
60FOR A=1 TO 1000:IF W=1 AND X=9 AND
Y=3 AND Z=4 T=W
70NEXT:PRINT(TIME-NOW)/100;" seconds"
80END
```

The IF series will work best if the the statement least likely to be true is placed first.

The use of IF in preference to AND for multiple conditions also has an advantage where a condition is not always valid. For example:

```
DIM A(100)
FOR I=1 TO 200
IF I<=100 IF A(I)=0 THEN . . .
```

will work, but:

```
DIM A(100)
FOR I=1 TO 200
IF I<=100 AND A(I)=0 THEN . . .
```

will crash when I>100 as Basic will attempt to evaluate both conditions, though the second is invalid for this value of I. The double IF format avoids this problem as the second condition is only evaluated when the first has already been found to be true. B

Points Arising.... Points Arising.... Points Arising....

BEEBUG DRAUGHTS
(Volume 7 No. 8 Page 55)

There are two minor errors in this program. Firstly, it is not possible to play any move ending on square A1, and secondly, the board is re-drawn incorrectly when a new game is started. These problems can be cured with the following changes:

```
2040 UNTIL ?list%>127
1505 VDU 29,0;0;
```

IMPROVED VIDEO CATALOGUER
(Volume 7 No.10 Disc & Cassette only)

The print option on this program will only work if line 5970 is removed. This has the side-effect that the output is echoed to the screen as it is printed.

PAGE COMPOSITION FOR THE BBC MICRO (2)
(Volume 7 No.10 Page 30)

There is an error in the font loading procedure in listing 2. Line 2540 should read:

```
2540 E%=OPENIN("Font"+STR$F%)
```

INDEXING DFS FORMAT DISCS
(Volume 7 No.10 Page 6)

This program will not index all the files on a disc as it stands. To correct this change the following line:

```
1140 J%=(names?5) DIV 8
```

B

POSTBAG

NARY A CROSS WORD

Now that I have SpellMaster installed in my Master 128, I do find that Keith Sumner's Crossword Editor (BEEBUG Vol. 7 No. 4) does appeal as promised, but I do have a problem.

When SpellMaster cannot find a word the program crashes, and the error "Not found at line 3260" appears. Can you explain how to keep the Crossword Editor running when this happens?

M.P.Ford

This is another instance of the limitations of error trapping in Basic on the BBC micro. When an error occurs within any loop, procedure or function, Basic loses all track of where in the program it is. This makes sensible recovery from errors very difficult, if not impossible, as here.

CUTTING CORNERS WITH GRAPHICS PRINTOUT

I have just bought your Best of BEEBUG Applications disc, and found all the programs excellent - with one exception. The Business Graphics program is just what I need for my work, but why is there no facility to print?

Is there some way of adapting the program so as to insert a screen dump, or to use a ROM such as Printmaster?

G.Lloyd Hughes

The Business Graphics program from the Best of BEEBUG Applications disc has a facility to save screen output to disc under the 'File' option in the main menu. This can then be printed out using any suitable printer ROM (such as BEEBUG's DumpMaster), or a screen dump program such as that published in BEEBUG Vol.5 No.5.

Note that the absence of the extreme bottom right-hand corner in any printout can be cured by changing '7FFF' in line 1960 of the Business Graphics program to '8000'.

CUTTING ASTAAD?

In BEEBUG Vol.7 Nos.5 to 7 there has appeared a new, superior version of ASTAAD, unfortunately (as far as I am concerned), only available on the Master series. Is there any way of recoding or reducing the program (with fewer features) such that the new version could be adapted to the model B?

David Wilson

The ASTAAD 3 drawing package was written specifically for a Master, and uses many of its features including sideways RAM, shadow RAM, and the extended plot functions and other features of Basic IV. It might be possible to modify the program to run on a model B fitted with additional RAM and the Acorn GXR extended graphics ROM, but there is no accepted standard for the use of sideways and shadow RAM add-ons, and ASTAAD.

Without additional memory, a cut-down version of ASTAAD 3 would probably be no more powerful than the original ASTAAD (which would probably be a better starting point than ASTAAD 3).

PROBLEMS WITH BEEBUG MENUS

I have found that the menu program on recent monthly magazine program tapes reports syntax errors and refuses to run correctly. In both cases the error lines contain the keyword EQUS. Can you suggest what the problem (and solution) is?

Michael Spearing

The problem is caused by changes which were made to the menu program from Vol.7 No.8 onwards. The new version uses assembler directives such as EQUS which are only recognised by Basic II, not Basic I. This applies to both tape and disc versions. Many of the programs now published in BEEBUG will not work with Basic I unless suitably modified.

To find out which version of Basic is in your machine, press Break and then type:

```
REPORT      <Return>
```

A copyright message will be printed, with the date 1981 indicating Basic I, and 1982 indicating Basic II.

Basic I has not been available since 1982, and Acorn stopped supporting it four years ago. BEEBUG also made the same move in June 1988 when Basic II ROMS became readily available, and this became fully operative from Vol.7 No.7 when we simplified our program icons. The Basic II ROM can be purchased from BEEBUG (stock code 0207A) for £20.76 (to members) plus 60p p&p.

Personal Ads

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

We also accept members' Business Ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, Dolphin Place, Holywell Hill, St. Albans, Herts AL1 1EX. The normal copy date for receipt of all ads will be the 15th of each month.

WANTED: Archimedes, preferably 310 but 305 considered. Tel. (0332) 556381.

WANTED: An upgrade kit to upgrade a Master 128 to a Master 512. Tel. 031-449 5308.

BEEBUG mag Nos. 5-10 vol 1, vols 2,3,4,5,6,7, offers. Acorn User magazine Feb 84 to Aug 88, offers, Micro User magazine Sept 84 to Dec 88, offers. Advanced Computer course two vols £10, Advanced User Guide (in binder) £10, assorted BBC books £10, cleaning disc in disc box £50. Tel. (0332) 566381.

BBC B 128k Sideways and Shadow RAM, double-sided twin disc drive, 1770 disc interface, £350 worth of software on tape and disc, cassette recorder, joystick, books and leads etc, will sell for £300. Tel. 01 801 9770 after 5pm.

Master 128 purchased Nov '88 including View, ViewSheet, Edit etc with Interword and Help II. Acorn colour monitor, as supplied compact, Archimedes. Cumana CS400S 40/80 disc drive with PSU. Care double ROM cartridge, Delta joystick, monitor plinth, dustcover, box of 10 5.25" discs, reference manuals 1 and 2, many issues Acorn User, BEEBUG, many games. All in 1st class cond. with full documentation key strips etc. Kit less than 5 months old, purchased from BEEBUG and still under warranty £550 o.n.o. Tel. (0283) 226271 eves, wk/ends.

BBC B 1.2 OS 40T disc drive, 32k Watford RAM, ATPL SWR board, AMX mouse, ROMs: Pagemaker, View, BEEBUG C Complier, Exmon II, Acorn GXR, Games £380 o.n.o. Tel. 01-635 8405 after 7pm.

OPUS double sided 40/80T dual drive model 5802 DB + manual £120. Tel. (0963) 33605.

BBC B software; Spellcheck £15, Wordease disc £10, Clares Replica III £5, Toolkit Plus ROM £20,BBC Advanced User Guide andDisc User Guide £5 each, Scrabble £5, also Interword ROM £25, ADFS Masterfile II £12, Dabs Viewsheet & Viewstore book and disc £18. All originals with manuals. Tel. (0788) 521189 after 5pm.

BBC compatible 5.25" D/S 80T disc drive £60. BBC PSU £40. Tel. 061 789 0500.

Archimedes 310M colour monitor + 20 discs, little used £950 o.n.o. or swap for Amstrad PC1640 with 32meg hard disc and ECD monitor. Also Watford Electronics Le Modem £45 and Morley Teletext adaptor and PSU £75. Tel. (0492) 593677.

Battery backed static RAMs. MK48208 equivalent to 6264. 200 nS, ideal for Master RAM cartridge or equivalent. With write protect switch set, will retain data for years even with computer switched off. Over £25 each new, sell £15. Tel. (0273) 723467 eves.

Technomatic Combo disc drive with 1 3.5" + 1 5.25" drive, mains powered £150. Tel. (0372) 743887 eves.

T.E.C. D/S disc drive 80/40T, home made case. £50 +p.p. Tel. (0768) 64890.

BBC B iss 7, with 1770 DFS, ADFS, and wordwise £300.Cumana twin40T disc drive S/S £125. Microvitec medium res. colour monitor £100.Apollo modem, chip and manual £50. Tel. (0733) 244682.

Minerva System Delta with card index and reporter boxed complete with manuals, discs and ROM £55. Beebugsoft Dumpmaster ROM £10. Beebugsoft CommandROM (not Hayes) £17. Beebugsoft Command ROM (Hayes) £17. Pace Linnet intelligent V21/23 modem (Hayes protocol) £75. Tel. (0734) 771230 day, (0734) 784897 eves.

Watford solderless ROM board with battery back up £18. Tel. (0823) 289378.

Master 512 dual 40/80 D/S disc drives, Nightingale modem with auto-dial board, Interword, Spellmaster, Master Genie, Floppywise Master, and various other ROMs, Peartree cartridges, 32k RAM and disc based software, Epson P4 parallel printer, VersEPROM, EPROM programmer, 100 discs and other extras. All manuals, leads etc included. All books hardware in A1 condition £650, will split. Also BEEBUG mags from issue 1 to date, in BEEBUG binders and mint condition. Offers please. Tel. (0782) 744531 ext. 3427 day, (0782) 771608 eves.

BEEBUGmagazines from vol.1 no.1 complete. £15 the lot! Tel. 061 881 0846 after 5pm.

Complete set of BEEBUG magazines in binders. Offers invited. Tel. (0923) 224548.

Archimedes 310M with RISC OS and PC emulator, less than one year old, 4way backplane/fan, external drive buffer. Green screen monitor, external 5.25" 40/80 D/S drive, tilt and swivel monitor stand. Some Arch. and PC software, virtually unused. £990 o.n.o. Tel. (0634) 241237.

Solidisk sideways RAM never used £20. WANTED: AMX orWatford mouse in good condition, willing to pay up to £20. Tel. (0224) 704677 after 5pm.

Master 1451 medium res. monitor, twin 40/80 D/S drives, BEEBUG Master ROM, ROM cartridge, copy holder, discs and discbox, many manuals books and magazines. Rarely used. £575 o.n.o. Tel. (0246) 416155 business hours.

Beebugsoft Discmaster £14, View printer driver generator disc £5, (both boxed and documented) Acornsoft Freefall disc £2, The Hobbit £4, Bandits at 3 o'clock £1, all suitable for the BBC B, the first two also Master compatible. Separately or the lot for £25. Tel. (0273) 832548.

Super Art £18, 3D Zicon £12, Dumpmaster £16, Printmaster £14, Toolkit £14, Teletext £8, Design £10, Superplot £7, Hershey Characters £10, LBO £14, View/Sheet guides £4, B Plinth £8, computer books, complete bound sets of BEEBUG and AU magazines offers. Tel. (0926) 335952.

Miracom/Miracle Modem WS3000 V2123. Unused £149 o.n.o. Tel. (0908) 6655803 eves.

Wordwise+ mint condition in original box £30. Tel. Ramsbottom 7710.

BBC B 40/80 drive, double plinth, 32k ROM board, Pagemaker, Superart, Mouse, Wordwise, Dumpout3, Music 500, over 100 games £300. Tel. 01-659 7763.

Cumana 3.5" disc drive, unused, double sided, no PSU, plus formatting disc £70. Tel. 01-806 6546.
BBC B issue 7 8271 DFS, plus over 50 original tapes £250. Master 128, Stop Press, Extra Extra, Viglen Cartridge system, 2 advanced reference manuals £250. DS40/80T PSU 800k Watford disc drive £100 plus much software and hardware, too numerous to list. Tel. (0226) 285172.

SWAP: Original Commstar 2 ROM with keystrip and documentation for 10 blank 3.5" discs or W.H.Y. Tel. (0532) 869425.

BBC B issue 7 software (cassette), tape recorder, leads £200 o.n.o. Tel. (0483) 576690.

WANTED: Quinkey five finger keyboard. Tel. 01-318 5155.

Electron computer, plus 1, Zenith 12" high res. green monitor and Brother M1009 printer. View and ViewSheet. All manuals £200. Tel. (04243) 3606.

ViewSheet £25, Slave ROM £15, Novacad with plotter driver £25, Commstar2 £20, Masterfile II £10. All complete packages as new. Various machine code programming books £3 each. Tel. (0736) 63918.

EXCHANGE: Have you upgraded from a BBC B to a Master 128 and wished you hadn't. I have a BBC B issue 7 with 8271 DFS and a Integra B expansion, I am interested in an exchange for a Master 128. Tel. 051 647 5367.

Aries B32 (shadow RAM) £48, Spellmaster £30, View 2.1, ViewSheet ViewStore £25 each, AMX Super Art £20, ViewSpell, GXR Graphics Ext., Toolkit Plus £15 each, Viewplot, ROMit, Sleuth, Exmon II, Printmaster £12 each, all in original package. Offers? Tel. (0372) 59530.

ROMit chip for BBC B £15. Tel. Bedford 49052.

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

		BEEBUG & RISC USER
£ 7.50	6 months (5 issues) UK only	£23.00
£14.50	1 year (10 issues) UK, BFPO, Ch.I	£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	

BACK ISSUE PRICES (per issue)

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

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.

Destination	First Item	Second Item
UK, BFPO + Ch.I	60p	30p
Europe + Eire	£1	50p
Elsewhere	£2	£1

POST AND PACKING

Please add the cost of p&p as shown opposite.

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: Glynn Clements
Advertising: Sarah Shrive
Production Assistant: Sheila Stoneman
Membership secretary: Mandy Mileham
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

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

Please submit your contributions on disc 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.

ISSN - 0263 - 7561

Printed by Newnorth-Burt Ltd (0234) 41111

Magazine Disc/Cassette

MAY 1989 DISC/CASSETTE CONTENTS

JULIA SETS - explore this extension of the Mandelbrot set with separate programs for Basic II and Basic IV.

SHARE INVESTOR - working program to assist the small investor in decision making on buying and selling stocks and shares.

COUNTING RABBITS - educational program aimed at very young children for use at home or school.

SPEED AND SOUND CONTROLLER - a most useful utility for the Master which allows the execution speed and sound of many programs to be controlled independently.

ANOTHER DATE WITH VIEW - a short utility which allows Master users to insert a permanent record of the current date into any text file.

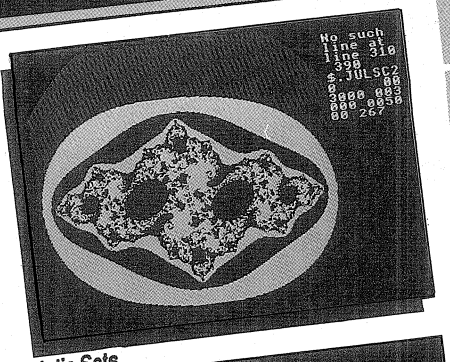
BIG TEXT SCREEN DISPLAYS - this utility automatically causes text displays to appear in double-height format in modes 0 to 6.

TRANSPARENT SIDEWAYS RAM LOADER - use this utility for the direct loading of sideways RAM images from disc.

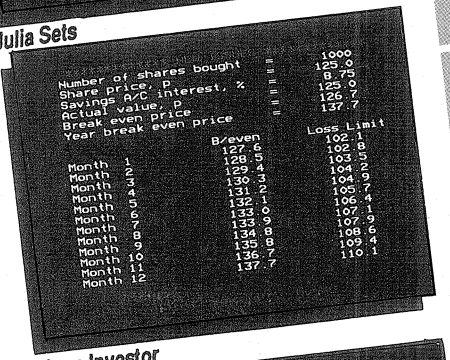
FIRST COURSE - four programs illustrating different techniques for the display of large-size digits on a mode 7 screen.

BOUNCER BALL - a superior implementation of a classic computer game which provides a testing challenge of your keyboard skill and concentration.

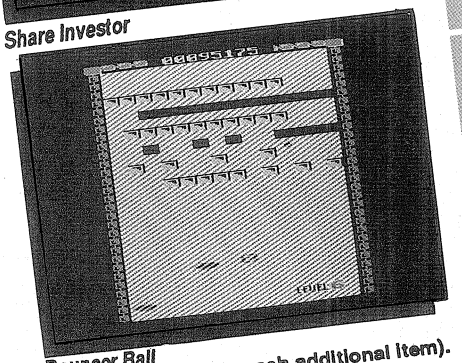
MAGSCAN - bibliography for this issue (Vol.8 No.1).



Julia Sets



Share Investor



Bouncer Ball

All this for £3. 50 (cassette), £4.75 (5" & 3.5" disc) + 60p p&p (30p for each additional item).
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	UK ONLY			OVERSEAS		
	5" Disc	3.5" Disc	Cassette	5" Disc	3.5" Disc	Cassette
6 months (5 issues)	£25.50	£25.50	£17.00	£30.00	£30.00	£20.00
12 months (10 issues)	£50.00	£50.00	£33.00	£56.00	£56.00	£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.

The NEW BBC Micro

The real information is out at last!

Acorn have recently announced details of the New BBC Micro. If you would like to know more, fill in the coupon below and return it to BEEBUG. If you prefer, call our 24 hour hot-line.

We've got the full details on the computer and will send you an info pack by return.

And if you're thinking of buying the New BBC Micro, not only can we offer a competitive deal but if you order now you'll be one of the first in the country to own the machine.

We also have a trade-in scheme to enable you to upgrade to the new computer. So if you have a BBC Micro, Master 128 or Compact we will take it in part exchange to reduce your outlay.

As a licensed credit broker we are also able to offer you 0% finance over nine months or flat rate 13.75% (typical APR 28%) over 12, 24 or 36 months.

DON'T DELAY, CONTACT BEEBUG TODAY.

Please send me information on the NEW BBC Micro

I am interested in 0% Finance 9 Months ❑

13.75% Finance over 12 months ❑

13.75% Finance over 24 months ❑

13.75% Finance over 36 months ❑

Trade-in BBC Micro etc. ❑

BEEBUG

Name _____

Address _____

_____

BEEBUG - THE RECOGNISED ARCHIMEDES SPECIALISTS

BEEBUG LTD., DOLPHIN PLACE, HOLYWELL HILL, ST.ALBANS, HERTS., AL1 1EX Tel. (0727) 40303