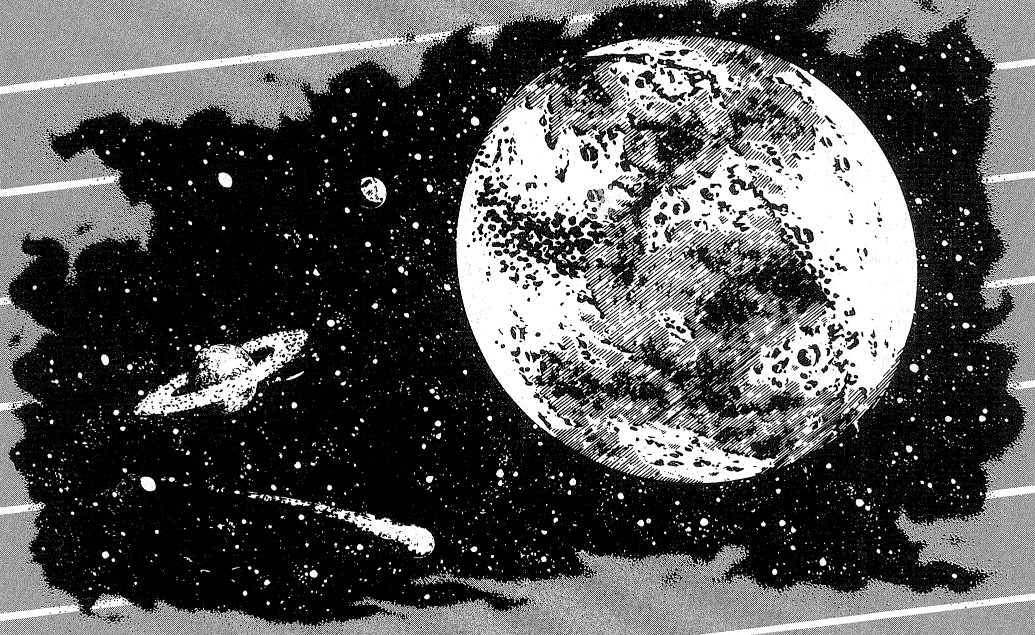
Vol.7 No.6 November 1988

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



Run the Solar System

● PROGRAM RESEQUENCER ● GAMES REVIEW

● MICROBRUSH REVIEW ● ROUTE PLANNER

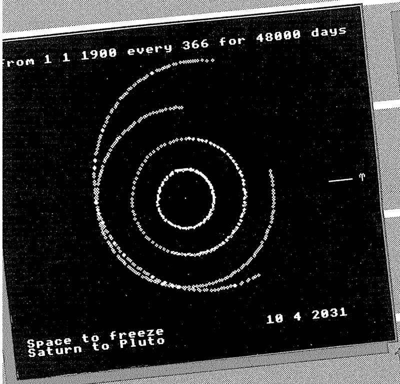
BEEBUG

Vol.7 No.6 November 1988

PROGRAM INFORMATION

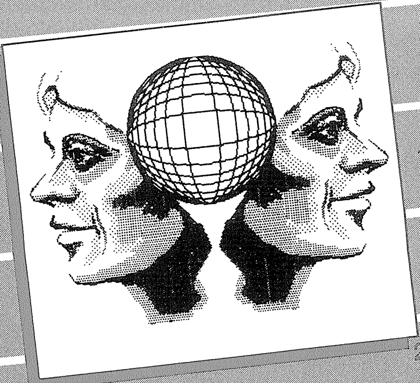
All programs listed in BEEBUG magazine are produced direct from working programs. They are listed in LISTO1 format with a line length of 40. However, you do not need to enter the space after the line number when typing in programs, as this is only included to aid readability. The line length of 40 will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

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

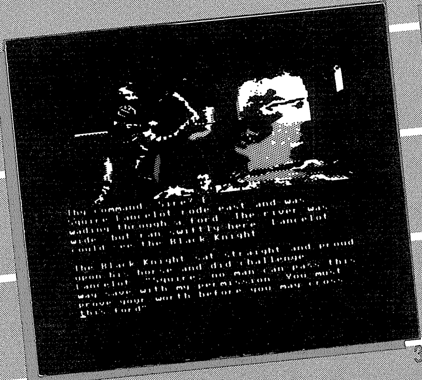


```
From 1 1 1900 every 366 for 48000 days

                                    10 4 2031
Space to freeze
Saturn to Pluto
```

1.

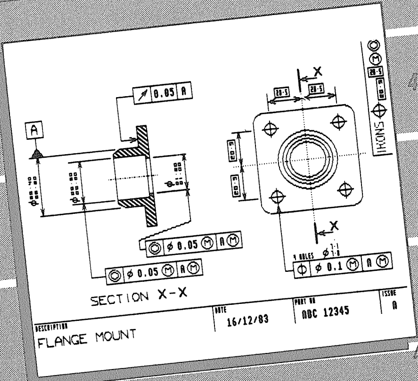


2.

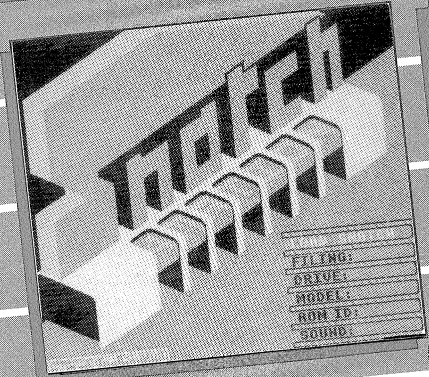


```
Thy command, Sire? E
Squire Lancelot rode east and was
wading through a ford. The river was
wide but ran swiftly here. Lancelot
could see the Black Knight.

The Black Knight sat straight and proud
upon his horse and did challenge
Lancelot: "Squire, no man can pass this
way save with my permission. You must
prove your worth before you may cross
this ford"
```

3.

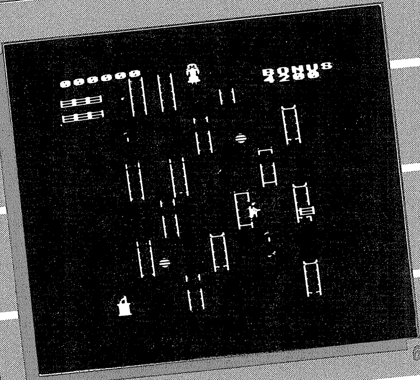


4.



```
FILING:
DRIVE:
MODEL:
ROM ID:
SOUND:
```

5.



6.

1. Run the Solar System

2. Microbrush Review

3. By Fair Means or Foul

4. Graphic Design with ASTAAD

5. Formation Snatching

6. Killer Gorilla

elow. These show clearly the valid combinations machine (version of Basic) and filing system for ach item, and Tube compatibility. A single line rough a symbol indicates partial working ormally just a few changes will be needed); a oss shows total incompatibility. Reviews do not stinguish between Basic I and II.

Computer System		**Filing System**	
Master (Basic IV)	M	ADFS	AS
Compact (Basic VI)	C4	DFS	DS
Model B (Basic II)	II	Cassette	▭
Model B (Basic I)	I	**Tube Compatibility**	
Electron	E	Tube	⊖

Editor's Jottings

Christmas is fast approaching and any orders for software or hardware should be placed as soon as possible to ensure early delivery. The accompanying BEEBUG Retail Catalogue is the Christmas edition, and we guarantee that all orders from this catalogue received by December 20th will be processed and goods dispatched (subject to stock still being available) in time to arrive before Christmas.

Our Technical Support Department will be pleased to help you with all your queries. Remember too, that we have a showroom - well worth visiting if you want help or advice, or to try out any item before you purchase. And as well as being open regularly every Thursday until 8pm, our showroom will also be open till the same time on Friday 23rd December, and till 5.30pm on Christmas eve for last minute purchases.

BUYING AN ARCHIMEDES

If you are considering buying an Archimedes, then it is well worth considering our 0% finance scheme. This takes only 48 hours to arrange. All the details can be taken down over the phone, documents can be posted to you to sign and return, and goods dispatched within two days of receipt. And you can still trade in your BBC micro or Master in part exchange.

BEST OF BEEBUG

An excellent and low-cost item for BBC micro and Master users is one of the Best of BEEBUG discs which we announced last month. With a wealth of quality programs from previous issues of BEEBUG magazine, both discs provide excellent value for money, and both discs are in stock for immediate dispatch. See the supplement pages for full details of these ideal stocking fillers.

RISC USER

Our magazine and support group for the Archimedes has recently celebrated the first anniversary of its launch. To mark this event we produced the RISC User Volume One Special Disc at a bargain price to members. This offer will only be valid until the end of December 1988. If you have an Archimedes and want a copy of this disc then join RISC User now at the reduced rate for BEEBUG members, and order your copy at the same time. In any case, a reduced rate subscription to RISC User has to be the best way of keeping up to date with the Archimedes world. See the supplement for more information on the RISC User Special Disc, and the inside back cover for reduced subscription rates for RISC User to BEEBUG members.

This month's telesoftware password is *ferryboat*.

News News News

MUSIC TO YOUR EARS

Hybrid Technology, producers of the *Music 5000* system, are about to launch the *Music 5000 Junior*. Although the new system is really a cut- down version of the *Music 5000*, it is designed to be an introduction for non-musicians. The software supplied with the *Music 5000 Junior* uses hi-res colour graphics and is fully icon driven. Tunes are entered using the keyboard, and you can easily move backwards and forwards through the score. All this software is supplied on a 32K PAL-PROM. Unfortunately, the *Music 5000 Junior* is not compatible with Ample music programs, or with the *Music 4000* keyboard, however, it is thought that a software upgrade will be made available at a later date. The price of the *Music 5000 Junior* has yet to be decided, but it will be under £100. More details can be obtained from Hybrid Technology, Unit 3, Robert Davies Court, Nuffield Road, Cambridge CB4 1TP, tel. (0223) 316910.

COMPILERS GALORE

Two separate Basic compilers for the Archimedes have been released in the space of a few weeks. The first was *ABC* (Archimedes Basic Compiler) from Dabs Press, which was quickly followed by *RiscBASIC* from Silicon Vision Ltd. Both compilers are similar, supporting all of the Basic language with very few restrictions,for example the EVAL statement, and both produce stand-alone ARM machine code. *RiscBASIC* and *ABC* cost £99.95 inc. VAT each. Silicon Vision has also launched *RiscFORTH*, a multi-tasking implementation of the FORTH-83 language. This also retails for £99.95 inc. VAT. Dabs Press can be found at 5 Victoria Lane, Whitefield, Manchester M25 6AL, tel. 061-766 8423, and Silicon Vision at Signal House, Lyon Road, Harrow, Middlesex HA1 2AG, tel. 01-427 5169.

TRANSFERRING TO THE MAC

Human Computer Interface Ltd, the company that is headed by former Acornsoft boss David Johnson-Davies, has extended its range of software for connecting the Beeb to an Apple Macintosh. Two packages are now available, *View>>Mac* for transferring View wordprocessor documents, and *Screen>>Mac* for the transfer of screens. *View>>Mac* allows text files to be transferred from the Beeb either as pure ASCII or View documents. In the latter case, all the View rulers, formatting and highlights (including italics etc.) are maintained, and the resulting document can be loaded straight into MacWrite. It is also possible to import the document into desktop publishing packages such as Aldus PageMaker and Quark Xpress. HCI Ltd. are currently working on a similar package for Wordwise Plus users. *Screen>>Mac* on the other hand allows the transfer and conversion of any Beeb screen (including mode 7) to the Macintosh's MacPaint or standard PICT formats. Grey scales are used to represent colours, and these can be edited from within the package. It is also possible to transfer screens back from the Mac to a Beeb. The packages cost £67.85 inc. VAT each, with the necessary serial link cable costing £33.35 inc. VAT. Further details can be obtained from Human Computer Interface Ltd., 25 City Road, Cambridge CB1 1DP, tel. (0223) 314934.

BUDGET GAMES

CDS Software are re-releasing fifteen classic Superior Software games in time for the expected Christmas rush. The games, which will include such greats as *Repton*, *Codename: Droid*, and *Deathstar*, will all be at a budget price of £2.99 each. Superior Software say that they are more concerned with the development of new games, and have therefore allowed CDS to release the old favourites, which will hopefully be stocked by high street multiples such as W.H.Smith.

AMSTRAD MODEM

As part of Alan Sugar's pledge to launch affordable high speed modems, Amstrad is about to release the *SM2400*. This modem supports four different standards, V21 (300 baud), V22 (1200 baud), V22bis (2400 baud) and V23 (1200/75 baud Prestel). The *SM2400* is fully Hayes compatible, which means that all its features, including auto-dial and auto-answer, can be used directly with software such as BEEBUG's Command ROM or BBC Soft's Modem Master, or with packages such as BEEBUG's Hearsay on the Archimedes. The price of the *SM2400* is £286 inc. VAT, which while being much dearer than bottom of the market budget modems, is very reasonable for a high-speed modem of this type. The *SM2400* should be available from most computer suppliers.

ECONET '88 SUCCESS

Once again, Acorn's annual conference for network users, which was held last month in Birmingham, attracted a large number of people. Acorn used the event, which was mainly attended by educational advisors, to launch its range of Stacking Filestores. The new Filestore, which is a complete Econet fileserver, differs from the original Filestore in that extra hard disc units can be added to increase the available storage up to a maximum of 240Mbyte. Acorn's Econet expert, Lawrence Hardwick, also talked about the future of Econet, and the use of Rank Xerox's Ethernet with the Archimedes, while Kim Spence-Jones of SJ Research revealed a new fileserver based around an 80386 IBM PC compatible.

LOCK UP THOSE PICTURES

Interactive Media Resources (IMR) has launched a new Genlock system for the Master 128. The *GL2000P*, as the system is called, allows graphics from the computer's display to be mixed with a video signal, and the results displayed on a monitor, or recorded back to video tape. This allows various effects to be achieved, such as adding credits to home movies. The *GL2000P* costs £805 inc. VAT. IMR has also produced a high-quality PAL encoder for the Master 128 and Archimedes, which allows the computer output to be recorded on video tape. This sells for £288 inc. VAT. Further details can be obtained from Interactive Media Resources, 8 North Street, Wolverhampton, West Midlands WV1 1RD, tel. (0902) 25444.

Run the Solar System

Donald Tattersfield presents his computerised orrery: a program that allows you to see the movement of planets around the sun.

The solar system consists mainly of the Sun around which 9 planets, Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune and Pluto respectively revolve. Our Earth has a very prominent moon and some of the other planets have their own moon or moons. There are also minor planets, comets and meteors. In this program we shall be concerned with the Sun and the 9 planets.

To the naked eye Mars, Jupiter and Saturn are visible in the night sky, while Mercury and Venus, if on view, can be observed in either the early morning or late evening sky. Of course binoculars or a small telescope considerably enhance their appearance. To the casual observer it is usual for only one or two planets to be seen in the sky at the same time, and these appear to move against the background of stars in a rather confusing manner so that the true, ordered, movement of the solar system is hidden.

THE PROGRAM

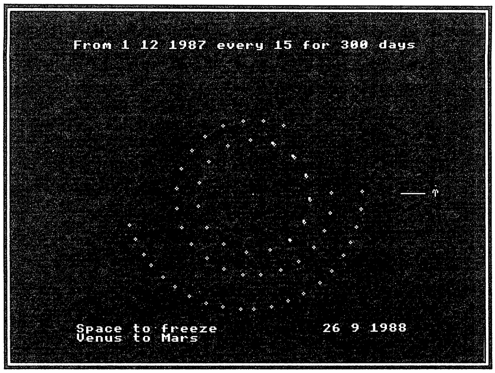
Type in the program and save it. When you run the program you will be asked to select an inner planet and an outer planet. The display will cover these planets and all those between them. If you select the same inner and outer planet you will see the motion of that planet only. The display is scaled so that the orbit of the outer planet always occupies the full screen. The program will then request the starting date for the display sequence. If the date supplied is after 1582 October 15 you will be working on the present everyday (Gregorian) calendar. Before that date the way of determining leap years was different, and you will be working on the Julian calendar. (Countries switched calendars at different times, hence 1582 will not apply to all countries.)

The next request will be for the time span you wish to cover in the display. Look at the DATA statements in lines 1000-1350. The data comes in 9 groups of 8 items, one group for each planet. After the name of the planet the first figure in each group is the number of years which the planet takes to travel once round the Sun. Thus to show one complete orbit for Mercury you will need to enter at least 88 days, or 90000 for Pluto. The positions of any planet in its second and subsequent orbits are marked by a symbol different from that of the first. If you input a positive value for the time interval when requested, you will see the planets move anticlockwise around the Sun, which is the correct direction as seen by an observer on the north side of the plane of the orbit of the Earth (known as the ecliptic plane). Supplying a negative time interval will move back in time.

The basic information which you have typed in will be displayed continually to remind you of your starting conditions, while the date of the latest displayed position of the planets can be found at the bottom right of your screen. Because mode 1 supports only a limited number of colours the sequence of colours is repeated from the inner to the outer parts of the solar system. The display can be frozen at any time by pressing the space bar. A further press will re-start the display.

MORE ABOUT THE PROGRAM

Apart from giving you a sense of power to be able to start and stop the solar system at your command, the program has an educational value and a practical one. What other information can you glean? First you will notice that some planets appear to move in circles while others, in particular Mercury and Pluto, have decidedly oval orbits. In fact all the orbits are ellipses with the Sun at one focus, but the eccentricity of most is very small - the second figure in each group of data. (A circle is an ellipse with zero eccentricity.) Then you will notice from the more elliptical orbits that the spacing of the successive positions of the planet is greatest when a planet is nearest the Sun, i.e. it is moving more quickly. These two features are a demonstration of Kepler's first two laws of planetary motion.



```
From 1 12 1987 every 15 for 300 days

Space to freeze               26 9 1988
Venus to Mars
```

Mars' close approach to Earth

Next you will see, on the right of your screen, a short line with the astronomical symbol for the First Point of Aries, the datum from which astronomical measurements are made. The anticlockwise angle between this line and the line from the Sun to a planet is the ecliptic longitude of the planet at that time (lr(Q) radians in the program). Notice that for a date very close to 21 March each year the direction of the Sun as seen from the Earth is the First Point of Aries. This is how the vernal equinox is defined (a date which has caused some disturbance at Stonehenge lately!). Similarly at the autumnal equinox, on or about the 21 September, the Earth is at the diametrically opposite point in its orbit.

We can also gain some idea about the scale of the solar system if we compare the orbits of some of the planets with that of the Earth whose Sun-Earth distance represents about 93 million miles (150 million km). I have said that the planets are sometimes not visible. If you choose a date when a planet is behind the Sun as seen from the Earth, you will, of course, not see it because of the glare of the Sun. You can observe the dates when this occurs from the program display. Also, if the Earth-planet line is on the left of the Earth-Sun line the planet might be seen in the evening sky, and conversely if on the right, in the morning sky. Readers should be aware that this is only one condition - there might be others which make the planet not visible, such as the geographical location of the observer.

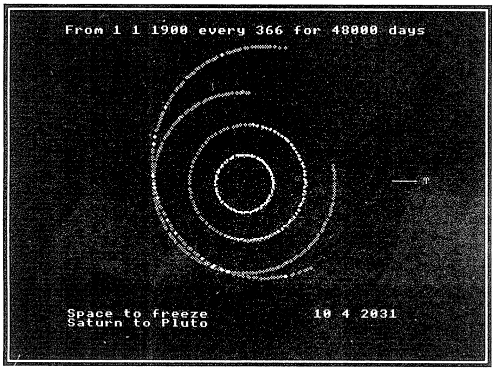
One final astronomical point - the fourth piece of data in each group gives an approximation to the greatest distance from the Sun which a planet achieves (the semi-major axis of its orbit). That for Pluto is greater than that for Neptune, so you might expect the orbit of Pluto to lie wholly outside that of Neptune. However, due to the inclination of the plane of the orbit of Pluto to the ecliptic plane (the sixth piece of data) Pluto actually moves inside the orbit of Neptune - as it is now and will be for some years. You can run the solar system for planets 8 and 9 and watch this happen, and determine the dates of cross-over. There is no likelihood of collision!- the orbits of all the planets have been projected onto the ecliptic plan for the display. You can also see why in September Mars was at its most brilliant for many years.

Because the planetary computations are somewhat involved, the smaller the number of planets chosen for the display, the quicker the program will run. The speed however is not affected by your choice of interval, nor by choosing inner planets rather than outer planets. The relative speeds of the planets always remains correct.

PROCEDURES

PROCnormalise simply brings an angle outside the range 0-360 degrees back into the range 0-360. PROCjulian converts the calendar date into Julian Date (not the same as Julian calendar date!), which is the number of days which have elapsed since 4713 BC. It is useful in many fields, as here, if you wish to know how many days there are between two calendar dates - convert each to Julian Date and subtract. PROCcalendar finds the calendar date from a Julian Date. The figure 2299160 in line 1430 is the Julian date of the changeover from the Julian to the Gregorian calendar, as used by this program.

You will see that the program gives you more (as they say in the adverts!) if you use it correctly. Astronomers will recognise the other pieces of data as the other parameters of the orbits. The third is the argument of perihelion, the fifth the longitude of the ascending node, and the seventh the longitude of the planet at the epoch (which in this case has been taken as 1975.0) - all in degrees, but you do not have to know about these items to enjoy the program.



```
From 1 1 1900 every 366 for 48000 days

Space to freeze                  10 4 2031
Saturn to Pluto
```

The orbit of Pluto inside that of Neptune

```
  10 REM >Solarsys
  20 REM Program Solar System
  30 REM Version B1.3
  40 REM Author  Donald Tattersfield
  50 REM BEEBUG  August 1988
  60 REM Program subject to copyright
  70 :
 100 ON ERROR IF ERR=17 THEN 920 ELSE V
DU4:PRINT TAB(0,30);:REPORT:PRINT" @ lin
e ";ERL:END
 110 MODE 7
 120 t$=STRING$(18,"* "):fact=360/365.2
5
 130 PRINT TAB(2,8) CHR$131 t$
 140 PRINT TAB(2,18) CHR$131 t$
 150 PRINT TAB(14,10);"SOLAR SYSTEM"
 160 PRINT TAB(7,15);CHR$134;"by Donald
 Tattersfield"
 170 TIME=0:REPEAT UNTIL TIME>200
 180 DIM P$(9),P(9),CY(9),W(9),AM(9),O(
9),S(9),ep(9),l(9),nu(9),r(9),dr(9),sr(9
),pr(9),lr(9),rd(9),x(9),y(9),R(9),N(9),
Lr(9)
 190 REM DEFINE PLANET AND SUN SYMBOLS
 200 VDU 23,240,192,192,0,0,0,0,0,0
 210 VDU 23,241,64,160,64,0,0,0,0,0
 220 VDU 23,242,128,0,0,0,0,0,0,0
 230 REM *FIRST POINT OF ARIES SYMBOL*
 240 VDU 23,243,0,20,42,42,8,8,8,0
 250 FOR Q=1 TO 9
 260 READ P$(Q),P(Q),CY(Q),W(Q),AM(Q),O
(Q),S(Q),ep(Q):NEXT Q
 270 MODE 7
 280 FOR Q=1 TO 9
 290 PRINT TAB(((Q-1)MOD3)*11,(Q-1)DIV3
);Q;"="P$(Q);:NEXT Q
 300 PRINT'TAB(0,4)CHR$133;"* SELECT IN
NER PLANET *        "
 310 INPUT TAB(25,4)inner%:IF inner%<0
OR inner%>9 THEN 300
 320 PRINT'TAB(0,6)CHR$133;"* SELECT OU
TER PLANET *        "
 330 INPUT TAB(25,6)outer%:IF outer%<0
OR outer%>9 THEN 320
 340 CC=180/PI:twoPI=2*PI:s$=" "
 350 PRINT'CHR$130"INPUT STARTING TIME"
'CHR$130"FOR THE SOLAR SYSTEM "
 360 INPUT'"YEAR (eg 1988 not 88) ",Y
 370 INPUT "MONTH (1-12)",M
 380 INPUT "DAY (1-31)",J
 390 PRINT'CHR$131"HOW MANY DAYS DO YOU
 WISH"'CHR$131"THE SOLAR SYSTEM TO RUN"
 400 INPUT dy
 410 PRINT'CHR$131"CHOSEN TIME INTERVAL
 (Negative"'CHR$131"goes back in time) (
Days)"
 420 INPUT dJ
 430 MODE 1
 440 REM * REDEFINE RED TO BE GREEN *
 450 VDU 19,1,2;0;
 460 PRINTTAB(0,0)"From ";J;s$;M;s$;Y;"
 every ";dJ;" for ";dy;" days";
 470 PRINT TAB(0,30)"Space to freeze";:
VDU5
 480 K=0
 490 REM * MAIN LOOP *
 500 REPEAT:GCOL 0,3
 510 PROCjulian:d=(I-2442412)+(D-0.5)
 520 PROCcalendar
 530 FOR Q=inner% TO outer%
 540 N(Q)=FNnormalise(fact*(d/(P(Q))))
 550 l(Q)=FNnormalise(N(Q)+(360*CY(Q)/P
I)*SIN((N(Q)+ep(Q)-W(Q))/CC)+ep(Q))
 560 nu(Q)=l(Q)-W(Q)
 570 r(Q)=AM(Q)*(1-CY(Q)*CY(Q))/(1+CY(Q
)*COS(nu(Q)/CC))
 580 dr(Q)=(l(Q)-O(Q))/CC:sr(Q)=S(Q)/CC
 590 pr(Q)=ASN(SIN dr(Q)*SIN(sr(Q)))
 600 Lr(Q)=ACS(COS(dr(Q))/COS(pr(Q)))
 610 IF pr(Q)<0 THEN Lr(Q)=twoPI-Lr(Q)
 620 IF pr(Q)=0 THEN Lr(Q)=dr(Q)
```

```
  630 lr(Q)=Lr(Q)+O(Q)/CC
  640 rd(Q)=r(Q)*COS(pr(Q))
  650 NEXT Q
  660 REM * INSERT THE SUN *
  670 GCOL0,2:MOVE600,500:PRINT CHR$242
  680 REM *INSERT FIRST POINT OF ARIES*
  690 MOVE 1100,500:DRAW 1180,500
  700 MOVE 1200,516:PRINT CHR$(243)
  710 MOVE 0,30
  720 PRINT P$(inner%)" to "P$(outer%);
  730 FOR Q=inner% TO outer%:GCOL 0,Q
  740 IF Q>3 THEN GCOL 0,Q-3
  750 IF Q>6 THEN GCOL 0,Q-6
  760 R=400/AM(outer%)
  770 R(Q)=R*rd(Q)
  780 x(Q)=R(Q)*COS(lr(Q)):y(Q)=R(Q)*SIN
(lr(Q))
  790 MOVE 600+x(Q),500+y(Q)
  800 IF dJ<=0 THEN 830
  810 IF K*dJ>P(Q)*365 PRINT CHR$240 ELS
E PRINT CHR$241
  820 GOTO 840
  830 IF dJ<0 AND K*dJ>-P(Q)*365 PRINT C
HR$241 ELSE PRINT CHR$240
  840 NEXT Q
  850 REM * PRINT THE DATE *
  860 VDU4:PRINT TAB(26,30);DD;s$;MM;s$;
YY;SPC3;:VDU5
  870 IF INKEY(0)<>32 THEN 900
  880 VDU4:PRINT TAB(0,30)"Space to re-s
tart";:REPEAT UNTIL GET=32
  890 PRINT TAB(0,30)"Space to freeze  "
;:VDU5
  900 K=K+1:J=J+dJ:V=K*dJ
  910 UNTIL V>dy OR (dJ<0 AND -V>dy)
  920 VDU4:INPUT TAB(0,2)"ANOTHER DISPLA
Y (Y/N)",Q$:VDU5:Q$=CHR$(ASCQ$ AND 223)
  930 IF Q$<>"Y" VDU4:END ELSE 270
  940 :
 1000 DATA Mercury
 1010 DATA 0.2409,0.2056,77.0665
 1020 DATA 0.3871,48.0349,7.0043
 1030 DATA 320.6631
 1040 DATA Venus
 1050 DATA 0.6152,0.006785,131.2193
 1060 DATA 0.7233,76.4547,3.3944
 1070 DATA 310.9745
 1080 DATA Earth
 1090 DATA 1.00004,0.01672,102.5104
 1100 DATA 1.0000,0,0
 1110 DATA 99.5343
 1120 DATA Mars
 1130 DATA 1.8809,0.09338,335.5988
 1140 DATA 1.5237,49.3647,1.8498
```

```
 1150 DATA 249.6292
 1160 DATA Jupiter
 1170 DATA 11.8622,0.04860,13.9199
 1180 DATA 5.2028,100.1961,1.3045
 1190 DATA 355.2141
 1200 DATA Saturn
 1210 DATA 29.4577,0.05563,92.5583
 1220 DATA 9.5384,113.4384,2.4893
 1230 DATA 104.1728
 1240 DATA Uranus
 1250 DATA 84.01247,0.04725,170.2547
 1260 DATA 19.1819,73.8728,0.7732
 1270 DATA 205.7829
 1280 DATA Neptune
 1290 DATA 164.7956,0.008586,44.4059
 1300 DATA 30.05796,131.5051,1.7724
 1310 DATA 249.9146
 1320 DATA Pluto
 1330 DATA 246.378,0.2461,224.2580
 1340 DATA 39.29976,109.9965,17.14451
 1350 DATA 202.3345
 1360 :
 1370 DEF FNnormalise(A)=A MOD 360
 1380 :
 1390 DEF PROCcalendar
 1400 JD=INT(I+D)+0.5
 1410 II=INT(JD+0.5)
 1420 FF=JD-INT(JD)
 1430 IF II<2299160 THEN 1460
 1440 AA=INT((II-1867216.25)/36524.25)
 1450 BB=II+1+AA-INT(AA/4):GOTO 1480
 1460 AA=1
 1470 BB=II+1+AA-INT(AA/4)
 1480 CD=BB+1524
 1490 DZ=INT((CD-122.1)/365.25)
 1500 EE=INT(365.25*DZ)
 1510 GG=INT((CD-EE)/30.6001)
 1520 DD=CD-EE+FF-INT(30.6001*GG)-0.5
 1530 IF GG<13.5 MM=GG-1 ELSE MM=GG-13
 1540 IF MM<2.5 YY=DZ-4715 ELSE YY=DZ-47
16
 1550 ENDPROC
 1560 :
 1570 DEF PROCjulian
 1580 IF M>2 THEN 1610
 1590 I=365*Y+INT((Y-1)/4)-INT((Y-1)/100
)+INT((Y-1)/400)+1721059
 1600 D=31*(M-1)+J+0.5:GOTO 1630
 1610 I=365*Y+INT(Y/4)-INT(Y/100)+INT(Y/
400)+1721059
 1620 D=31*(M-1)-INT((M-1)*.4+2.7)+J+.5
 1630 I=I+INT(D):D=D-INT(D)
 1640 PRINT':ENDPROC
 1650 ENDPROC
```

A Motorist's Route Planner

Barry Thorpe's neat program will help you to plan your next long distance motoring trip, and many more besides.

INTRODUCTION

When you are setting out on a long drive, especially if the route is not familiar, it is worthwhile spending some time working out a route plan. The program MOPLAN listed here will generate a route schedule showing estimated timings, and, if desired, suggested break times. The data required is derived from a text file created with any suitable word processor (see separate article in this issue). The output is designed to be sent to any suitable Epson compatible printer, but the program could be modified to provide an alternative screen display. This program was used to plan a journey from south of Manchester to Skye last year, and even I was impressed how closely the plan matched our actual journey; we arrived some fifteen minutes early.

The accompanying sample output below shows a plan for the journey from St Albans in Hertfordshire to St Ives in Cornwall. On the next page is the data file with a line-by-line explanation.

USING THE PROGRAM

Enter your wordprocessor or text editor (N.B. the output must be a pure ASCII file) and type in the details that suit your car and driving style. Press Return after each item. The sample file is clear on most points, but a few additional comments may be useful.

1. The data must be in the order stated.

2. The specification for breaks - in the example, an hour and three quarters or so between breaks, and three breaks of a half, one and a half, and half an hour respectively - may be replaced by a single asterisk, to prevent the program inserting breaks (as in the example).

3. The mpg figure is adjusted in the program according to the road type specified for that leg of the journey. If you want to fine-tune the mpg figures, the adjustments should be made in line 1820 in the accompanying program.

4. The journey details start with the measured distance on the map (one of those little wheel gadgets is very useful), followed by a slash, a description of the next point in the journey (with road number after a colon), a further slash, and a

```
                                      ROUTE PLAN
                                  St Albans TO St Ives
starting at 0700
from                          to                         miles  miles   time   time    time     av.
                                                         per    cumul.  per    cumul.  from     speed
                                                         leg            leg            0700

 St Albans                    A41 jct: A5183-A405          7      7     0:21   0:21     7:21    20
 A41 jct: A5183-A405          M25 jct 17:A405              4     11     0: 4   0:25     7:25    50
 M25 jct 17:A405              M3 jct 2:M25                19     30     0:19   0:44     7:44    60
 M3 jct 2:M25                 M3 jct 8 Wheatsheaf Inn:M3  31     61     0:30   1:15     8:15    60
 M3 jct 8 Wheatsheaf Inn:M3   Amesbury: A30-A303          28     89     0:33   1:49     8:49    50
 **BREAK**                                                              0:30   2:19     9:19
 Amesbury: A30-A303           Mere: A303                  22    111     0:26   2:45     9:45    50
 Mere: A303                   S. Petherton: A303          28    139     0:42   3:27    10:27    40
 S. Petherton: A303           Ilminster A30 jct: A303     16    155     0:24   3:51    10:51    40
 Ilminster A30 jct: A303      M5 jct 30 (Exeter): A30     19    174     0:22   4:14    11:14    50
 **BREAK**                                                              1:30   5:44    12:44
 M5 jct 30 (Exeter): A30      M5 jct 31: M5                4    178     0: 4   5:48    12:48    60
 M5 jct 31: M5                Okehampton: A30             22    200     0:26   6:15    13:15    50
 Okehampton: A30              Launceston: A30             19    219     0:22   6:37    13:37    50
 Launceston: A30              Bodmin: A30                 20    239     0:30   7: 7    14: 7    40
 Bodmin: A30                  Carland: A30                19    258     0:28   7:36    14:36    40
 **BREAK**                                                              0:30   8: 6    15: 6
 Carland: A30                 Camborne: A30               17    275     0:20   8:26    15:26    50
 Camborne: A30                St Ives: A30-A3074          10    285     0:30   8:56    15:56    20

petrol used (rounded up): 8 gallons, costing £14.00 over 285 miles
```

```
DATA FILE EXPLAINED

M                                 unit code: miles (K for km)
3                                 map scale: 1" to 3 miles
20                                av. speed urban
40                                ditto outer suburbs etc
50                                ditto open country
60                                ditto motorways
40                                best mpg figure (adjusted in program)
1.75                              price petrol per gallon
St Albans                         start point
St Ives                           Cornwall destination
0700                              start time 4 digits 24 hr clock
1.75/0.5/1.5/0.5                  1.75 - max time before break: length of each break
2/A41 jct: A5183-A405/1            ****
1/M25 jct 17:A405/3               journey legs: first entry explained:
6/M3 jct 2:M25/4                       2 map inches (or cm) TO A41 jct on
10/M3 jct 8 Wheatsheaf Inn:M3/4        road type 1
9/Amesbury: A30-A303/3
7/Mere: A303/3                    7 map inches to Mere on road type 3
9/S. Petherton: A303/2
5/Ilminster A30 jct: A303/2
6/M5 jct 30 (Exeter): A30/3
1/M5 jct 31: M5/4
7/Okehampton: A30/3
6/Launceston: A30/3
6.5/Bodmin: A30/2
6/Carland: A30/2
5.5/Camborne: A30/3
3/St Ives: A30-A3074/1
Q                                 terminating value + RETURN
```

number representing the road type. This number is used by the program to estimate the time taken over that leg of the journey.

5. The last entry should be 'Q' and Return.

Save the file as an ASCII file. With Wordwise there is no problem; with Interword, turn paging off, set the ruler for zero left margin, and save with option 8 - spool with no codes; for View, just omit any rulers and embedded commands.

Assuming that you have typed it in and saved it, chain the program, which first reminds you of the data required. Insert the disc containing the data file in the appropriate drive and press Return. Enter the file name, and the program reads the file and interprets the details as far as the breaks. If, through some earlier mistyping, there is a screen full of garbage, you can stop at this point to correct the errors. If not, when you press "Y", the file will be processed immediately to produce the route plan. Any errors will be flagged on the printout.

PROGRAM NOTES

A number of printer control codes are included in the program using VDU statements (lines 1830, 1840, 1850 etc). Change or omit these as appropriate. The VDU 21 and VDU 6 codes which appear (lines 1830, 1880, 2050, 2080, 2390 and 2440) switch the VDU drivers on and off so that printer output is not displayed on the screen. Again, these codes could be omitted. The resulting display will be too wide for the screen in mode 7, but a change to mode 3 (using a new line 125) would overcome this, though the various teletext control codes would need to be removed to tidy up the screen. Happy motoring.

```
  10 REM Program MOPLAN
  20 REM Version B1.0
  30 REM Author  Barry Thorpe
  40 REM BEEBUG  November 1988
  50 REM Program subject to copyright
  60 :
 100 MODE7:ON ERROR GOTO 170
 110 PROCinit
 120 PROCintro
 130 PROCprocess
 140 VDU26,12
 150 END
 160 :
 170 MODE 7:VDU6:CLOSE#0
 180 REPORT:PRINT" at line ";ERL
 190 END
 200 :
1000 DEF PROCinit
1010 @%=10:*FX229,1
1020 DIM avspeed(4),estmpg(4),break(10)
1030 t1=5:t2=35:t3=70:t4=78:t5=89:t6=99
:t7=109:t8=120:
1040 ENDPROC
```

```
 1050 :
 1060 DEF PROCdouble(t$,bc%,fc%,ln%)
 1070 x%=(39-LENt$)DIV2-4
 1080 VDU31,x%,ln%,141,bc%,157,fc%:PRINT
 t$;SPC2;CHR$156
 1090 VDU31,x%,ln%+1,141,bc%,157,fc%:PRI
NT t$;SPC2;CHR$156
 1100 ENDPROC
 1110 :
 1120 DEF FNupper(s$)
 1130 LOCAL A%,B$,I%:B$=""
 1140 FOR I%=1 TO LENs$
 1150 A%=ASC(MID$(s$,I%,1))
 1160 IF A%>96 AND A%<123 THEN A%=A%-32
 1170 B$=B$+CHR$(A%)
 1180 NEXT
 1190 =B$
 1200 :
 1210 DEF PROCcentre(t$,y%)
 1220 len%=LEN(t$):x%=(39-len%)DIV2
 1230 PRINTTAB(x%,y%)t$;
 1240 ENDPROC
 1250 :
 1260 DEF PROCtestbuffer
 1270 REPEAT
 1280 *FX21,3
 1290 probe1=ADVAL(-4):VDU2,1,0,1,0,3:pr
obe2=ADVAL(-4)
 1300 printerconnected=probe1=probe2
 1310 IF NOT printerconnected THEN CLS:P
RINTTAB(7,10)CHR$129;CHR$136"Please conn
ect printer"''TAB(8)"Press Return to con
tinue":REPEAT UNTIL GET=13
 1320 UNTIL printerconnected
 1330 ENDPROC
 1340 :
 1350 DEF FNgetstring
 1360 LOCAL A%,S$:S$="":A%=BGET#X
 1370 REPEAT
 1380 S$=S$+CHR$(A%)
 1390 A%=BGET#X
 1400 UNTIL A%=13
 1410 =S$
 1420 :
 1430 DEF PROCintro
 1440 PROCdouble("MOTOR ROUTE PLANNER",1
30,132,0)
 1450 PROCcentre(CHR$134+"You need a tex
t file of this data",5)
 1460 RESTORE 1510:
 1470 FOR item=1 TO 10:
 1480 READ thing$
 1490 PROCcentre(CHR$(134-3*(item MOD2))
+thing$,item+6)
 1500 NEXT
 1510 DATA units (miles/km),map scale,4
average speeds,touring mpg figure,price/
gallon petrol,starting place,destination
,a starting time,breaks,intermediate pla
ces
 1520 PROCcentre(CHR$134+"Make sure the
printer is ready",item+7)
 1530 REPEAT UNTIL GET=13
 1540 PROCtestbuffer
 1550 PROCfilename
 1560 ENDPROC
 1570 :
 1580 DEF PROCprocess
 1590 CLS:PRINTTAB(10)CHR$131"Check file
 header:"':RESTORE 1510:
 1600 unit$=FNupper(FNgetstring)
 1610 IF unit$="M" THEN unit$="miles" EL
SE unit$="kms"
 1620 READ item$:PRINTitem$ TAB(25)unit$
 1630 scale=VAL(FNgetstring)
 1640 READ item$:PRINTitem$ TAB(25);"1:"
;scale
 1650 READ item$:PRINTitem$ TAB(25);
 1660 FOR type=1 TO 4
 1670 avspeed(type)=VAL(FNgetstring):PRI
NT;avspeed(type);" ";
 1680 NEXT
 1690 mpg=VAL(FNgetstring)
 1700 READ item$:PRINT'item$ TAB(25);mpg
 1710 pprice=VAL(FNgetstring)
 1720 READ item$:PRINTitem$ TAB(25);ppri
ce
 1730 start$=FNgetstring
 1740 READ item$:PRINT'item$'CHR$134 sta
rt$
 1750 dest$=FNgetstring
 1760 READ item$:PRINT'item$'CHR$131 des
t$'
 1770 startime$=FNgetstring:sth%=VAL(LEF
T$(startime$,2)):stm%=VAL(RIGHT$(startim
e$,2))
 1780 READ item$:PRINTitem$ TAB(25) star
time$
 1790 break$=FNgetstring:IF LEFT$(break$
,1)<>"*" THEN PROCdecodebreaks ELSE brea
k(0)=10E6
 1800 PRINTTAB(5,23)CHR$134"OK? Y/N":REP
EAT:A$=GET$:UNTIL INSTR("YyNn",A$)
 1810 IF INSTR("Nn",A$) THEN CLOSE#0:VDU
6:END
 1820 estmpg(1)=0.75*mpg:estmpg(2)=0.85*
mpg:estmpg(3)=mpg:estmpg(4)=0.8*mpg
 1830 VDU2,1,15,21,1,14:PRINTTAB(29)"ROU
TE PLAN "'
 1840 len%=LENstart$ + LENdest$ + 4:VDU1
,14:PRINTTAB((68-len%)DIV2) start$" TO "
dest$'
 1850 VDU1,14:PRINTTAB(t1)"starting at "
startime$'
 1860 PRINTTAB(t1)"from" TAB(t2)"to" TAB
(t3)unit$ TAB(t4)unit$ TAB(t5)"time"TAB(
t6)"time" TAB(t7-1)"time from "TAB(t8)"a
v.speed"
 1870 PRINTTAB(t3-2)"per leg" TAB(t4)"cu
mul." TAB(t5)"per leg" TAB(t6)"cumul." T
AB(t7+1)startime$'
 1880 VDU6,3
 1890 REM main loop starts here
 1900 gallons=0:cdist%=0:ctime=0:brktime
r=0:brk=1
 1910 sect$=FNgetstring
 1920 REPEAT
 1930 V%=INSTR(sect$,"/"):length=VAL(LEF
T$(sect$,V%-1)):valid=V%>0:
 1940 V%=V%+1
 1950 V2%=INSTR(sect$,"/",V%):valid=vali
d AND V2%>0:
 1960 point$=MID$(sect$,V%,V2%-V%)
 1970 type=VAL(MID$(sect$,V2%+1))
 1980 valid=valid AND (length>0)AND(type
>0)AND(type<5)
 1990 IF valid PROCcalcandprint ELSE PRO
Cerrorprint
 2000 IF brktimer>=break(0) THEN PROCins
ertbreak
 2010 sect$=FNgetstring
 2020 UNTIL LEFT$(FNupper(sect$),1)="Q"
 2030 CLOSE#X
 2040 gallons=1+INT(gallons):cost=pprice
*gallons
 2050 VDU2,21
 2060 PRINT''TAB(5)"petrol used (rounded
 up): ";gallons" gallons, costing `";
 2070 @%=&20206:PRINTcost "  over ";:@%=
10:PRINT;cdist% " "unit$
 2080 VDU6,3
 2090 ENDPROC
 2100 :
 2110 DEF PROCdecodebreaks
 2120 C%=0:S%=1:F%=INSTR(break$,"/",S%)
 2130 REPEAT
 2140 break(C%)=VAL(MID$(break$,S%,F%-S%
))
 2150 C%=C%+1:S%=F%+1
 2160 F%=INSTR(break$,"/",S%)
 2170 UNTIL C%>10 OR S%=1
 2180 numbrks=C%-1
 2190 PRINT"max driving period" TAB(25);
break(0)
 2200 PRINT"breaks of"TAB(25);
 2210 FOR C%=1 TO numbrks:PRINT;break(C%
)" ";:NEXT
 2220 ENDPROC
 2230 :
 2240 DEF PROCfilename
 2250 CLS
 2260 REPEAT
 2270 PRINTTAB(5)CHR$134"Which file ";:I
NPUT f$
 2280 X=OPENUP(f$)
 2290 IF X=0 VDU7
 2300 UNTIL X>0:
 2310 ENDPROC
 2320 :
 2330 DEF PROCcalcandprint
 2340 dist%=length*scale+1:cdist%=cdist%
+dist%
 2350 gallons=gallons+dist%/estmpg(type)
 2360 time=dist%/avspeed(type):ctime=cti
me+time:brktimer=brktimer+time
 2370 hrs%=INT(time):mins%=(time-INT(tim
e))*60:
 2380 PROCcalc_elapsedtime
 2390 VDU2,21
 2400 PRINTTAB(t1)LEFT$(start$,28) TAB(t
2)LEFT$(point$,28);:start$=point$
 2410 @%=4:PRINTTAB(t3)dist%;:@%=5:PRINT
TAB(t4)cdist%;
 2420 PROCprint_elapsedtime(hrs%,mins%)
 2430 PRINTTAB(t8) avspeed(type)
 2440 VDU6,3
 2450 ENDPROC
 2460 :
 2470 DEF PROCerrorprint
 2480 VDU2,21
 2490 PRINT sect$" <<<ERROR"
 2500 SOUND 1,-15,200,20:SOUND 2,-15,100
,20:
 2510 ENDPROC
 2520 :
 2530 DEF PROCinsertbreak
 2540 IF brk>numbrks THEN ENDPROC
 2550 bh%=INT(break(brk)):bm%=(break(brk
)-INT(break(brk)))*60:
 2560 ctime=ctime+break(brk)
 2570 PROCcalc_elapsedtime
 2580 VDU2,21
 2590 PRINTTAB(t1)"**BREAK**";:PROCprint
_elapsedtime(bh%,bm%)
 2600 PRINT
 2610 brktimer=0:brk=brk+1
 2620 VDU6,3
 2630 ENDPROC
 2640 :
 2650 DEF PROCcalc_elapsedtime
 2660 chrs%=INT(ctime):cmins%=(ctime-INT
(ctime))*60:
 2670 h%=sth%+chrs%:m%=stm%+cmins%
 2680 IF m%>59 THEN m%=60-m%:h%=h%+1
 2690 IF h%>23 THEN h%=24-h%
 2700 ENDPROC
 2710 :
 2720 DEF PROCprint_elapsedtime(hrs%,min
s%)
 2730 @%=2:PRINTTAB(t5)hrs%":"mins% TAB(
t6)chrs%":"cmins% TAB(t7)h%":"m%;
 2740 ENDPROC
```

MicroBrush from AB Designs

Reviewed by Roger Burg

Product	**Microbrush**
Supplier	**AB Designs**
	81 Sutton Common Road,
	Sutton, Surrey SM1 3HN.
	Tel. 01-644 6643
Price	**£150 (2-ROM package)**
	£200 (3-ROM package - needs sideways RAM)

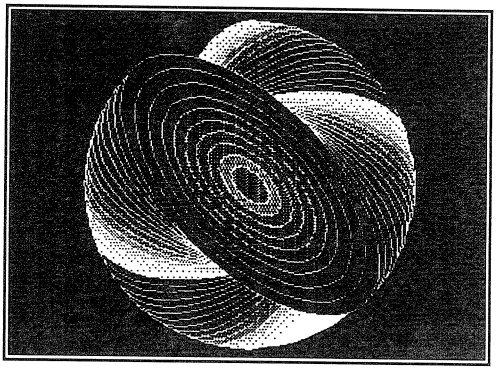
Microbrush is a most unusual package. I bought AB's first graphics program when it was the only one in existence for the BBC micro. Over the last five years it has been constantly refined and developed. A couple of years back, the package was re-written and re-released in ROM version as Microbrush, and these two ROMs (after two years of upgrading) are reviewed here. But there is a third ROM for the Master, and a fourth is about to be released.

Microbrush prepares text and images in modes zero and one. With the third ROM the image size can be many times greater than the screen size, using sideways RAM, and the printed output allows millions of colours, mixes and tints. A fourth and final ROM is due for release. Here I shall review only the two-ROM package.

Microbrush provides twelve groups of functions, called 'systems'. A second ROM contains many routines called from within Microbrush, but available to your programs too. Some invaluable extras are supplied on disc, and alternative palettes, fonts and screens are on two additional discs. The package can be expanded still further by adding your own routines, palettes, icons and special effects. The different sections are well integrated, and each deals with functions in its own area in detail.

The package provides the usual 'painting' facilities, and two different text systems: one using bit-mapped fonts, and the other using scalable, profile-defined fonts. Its wide range of dumps for colour and monochrome printers is exceptionally thorough. The quality of text, graphics and printer dumps makes this the only package I have seen for the BBC micro, where the output would not look out of place in a colour supplement, although it avoids the usual extravagant claims to being 'desk top publishing'.

Microbrush assumes that you have a pretty good monitor, but should you need it, a magnified pixel editor is accessible from the disc.



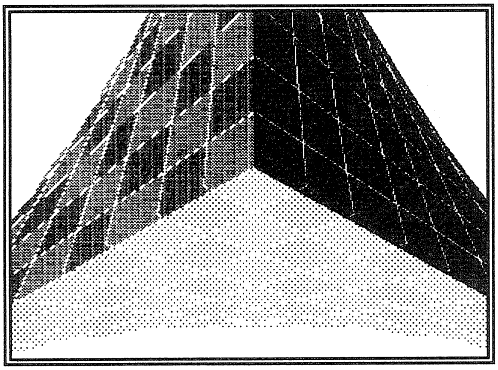
The system is designed to run with a tracker ball. This is a better graphics tool than a mouse, and the program takes full advantage of it. But, if you must use a mouse, it also provides the software locks to make it a more useful graphics tool. It is wonderful to have a still pointer when you press a button or set text, and the accuracy of finger-tip control is positively exhilarating, after the rodent epidemic. And of course the tracker ball is much more compatible with cow gum, pencil rubbings and tea cups.

Inevitably there is far more to a package which has been developed and used over five years than can be said in a review. To convey the idea of its scope, I shall list only the main categories of the two-ROM version, and then I devote a section to each of them.

OVERVIEW

The first ROM contains painting routines, the colour rubber, full-colour pattern-fill, line drawing routines, icon generation and display facilities, the typesetter and type generator, special effects generator, mode translation routines and a nice disc access system.

The second ROM contains most of the 'CAD' features, lines, rectangles, circles and polygons, all quick and uncommonly easy to place and adjust with great precision, and with the additional vector zoom facilities, routines to clip, squash, stretch, and mirror, all integrated well in the package. And it contains other, more specialised facilities which I can't describe concisely (a comprehensive perspective system which uses these shapes comes on ROM 3).

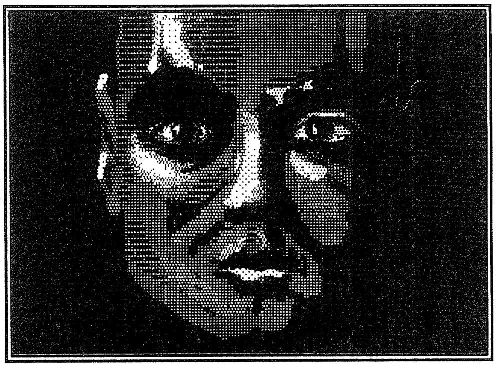


I shall concentrate on the painting system, the special effects generator and the screen dump facility.

PAINTING

At first, the painting system looks much like any other. You pick up pixel-mixed colours from a menu, and splosh them on the screen with various shaped and sized 'brushes', painting over colours, underneath them or with degrees of transparency. But the process contains more options than other packages, and is more effective.

I can't fathom the reason it works so well, but the program incorporates some original solutions to old problems of using pixel mixes. The mixes provide a useful range of colours which have proven to work well together over the years. That is unusual. But the program also attacks the problem of how to achieve the gradations of colour required.

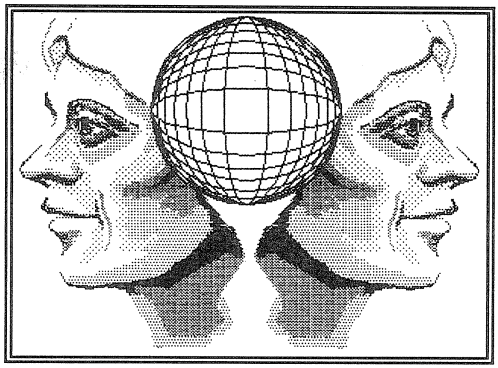


To cope with this, you can use pixel patterns which repeat at intervals of two to eight pixels, as required, both vertically and horizontally. This allows large areas to be covered evenly (and makes for some interesting programming problems). What amazed me, however, was the way in which the program can draw up these mixes, so that they can be over-painted by other colours. Algorithms for pixel-mixing the colours usually produce dreadful interference patterns (I once spent several weeks trying to produce an acceptable pixel-mix routine, and failed). But when one mix overlaps another, without introducing jagged patterns to advertise the joins, then the program is something very special.

One week ago, I would have smiled if someone had compared a mode one screen to oils and canvas, but using Microbrush felt strangely like drawing a good filbert across a rough ground. In fact, I finished a modeled face in less than an hour - even to get acceptable results in mode one was a new experience.

Like all good software, the program allows you to adjust the defaults. So the disc supplies alternative palettes, and the manual explains how you can build and save your own. If you create interesting mixes of colour, you can grab them to the palette or save them to disc. And if an important mix won't work because it is awkwardly out of register with the colours on screen, you can scroll it at the press of a key, until it fits.

The painting routines include facilities to paint with any shape, and to erase different colours and patterns. There is even a special, limited fill routine which will work in the pixel-mixed colours, including those which contain the logical background colour. So there's a lot more to tell.



PRINTER DUMPS

The printer routines are the most comprehensive I have seen. The monochrome, Epson compatible dumps come in eight varieties. Each has several settings, some of which permit wide variations. The image can be clipped, and dumped in a range of Epson ratios. The fashion for 'scaled' dumps (which really means rejecting or doubling crucial rows and columns of pixels) is avoided, and options concentrate on balancing the contrast and saturation of greys, achieving good saturated blacks and inverting tonal scales if required.

The routines allow the user wide adjustment of the grey scales, very fine tuning of the contrast scale, and a fool-proof test dump, to allow you to adjust for the age of the printer ribbon. I only regret that having fiddled in the past over dumps from other packages so that they printed respectably for publication, the scale of improvement in the present dumps may be hidden from the reader!

These are the only dump routines which I have come across which use over-printing to achieve the grey levels, instead of printing chequerboard patterns of mini-pixels to approximate to them. Colour separation is also provided. This is intended for commercial reproduction, but if you want to design in colour, and then print in three different coloured ribbons, you can do that too.

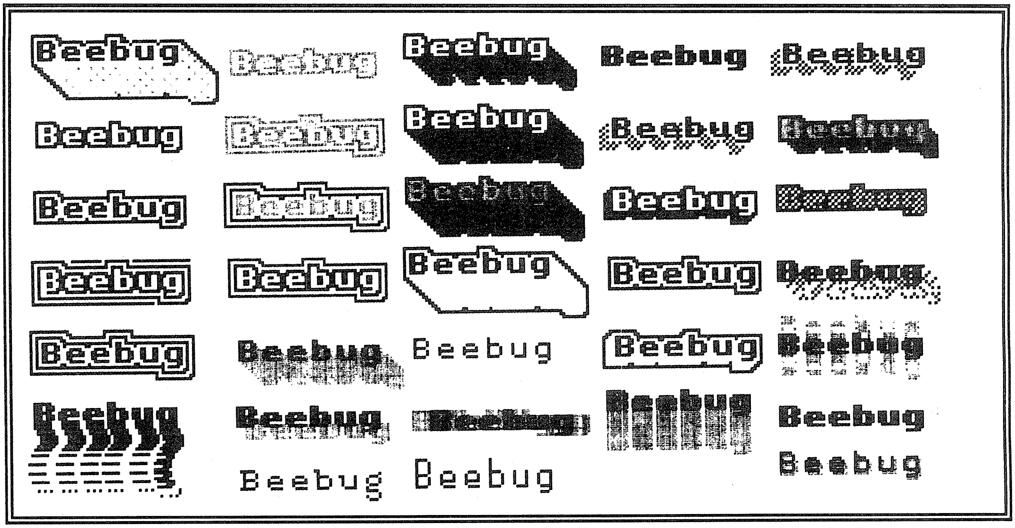
The routine omits three of the usual features - reporting if the printer is not receiving, using the usual BBC line-feed dip-switch setting, and taking default tone values from the 'actual colour' palette. With this quality of dump, the tones must be hand-tuned anyway, and these points may yet be modified if spare bytes can be found for them.

Of course, there is also a wide range of facilities for the Integrex colour printer, which produces remarkable results, and these, used with the super-sized mode 1 and mode 0 screens in the extra colours provided by the third ROM, surpass anything I expected to see on the BBC micro. I can do no more than mention them here, but they must be seen to be believed.

SPECIAL EFFECTS

I shall describe one more feature in some detail. The text facilities are too vast. There are high resolution, bit-mapped fonts with the 3-ROM version (or they can be bought as an extra disc), and low resolution ones which are provided in the basic package. Of course you can define your own, and the utility for doing this shows practical insight into the way the initial shapes can be knocked up quickly.

and it pops up, inverted, with outline and automatic drop shadow. But then you can go on to add your own effects to the function keys, and save them on disc for later use.



In addition to that, the zoom-shape system supports a range of large fonts defined by outline. And it goes without saying that they are proportional, with optional manual adjustment from the tracker ball. And, if you want some visual fireworks, then you can create logos, or new flashy fonts in the special effects generator. In fact, typical of the integration of the package, this facility can be used on any other image.

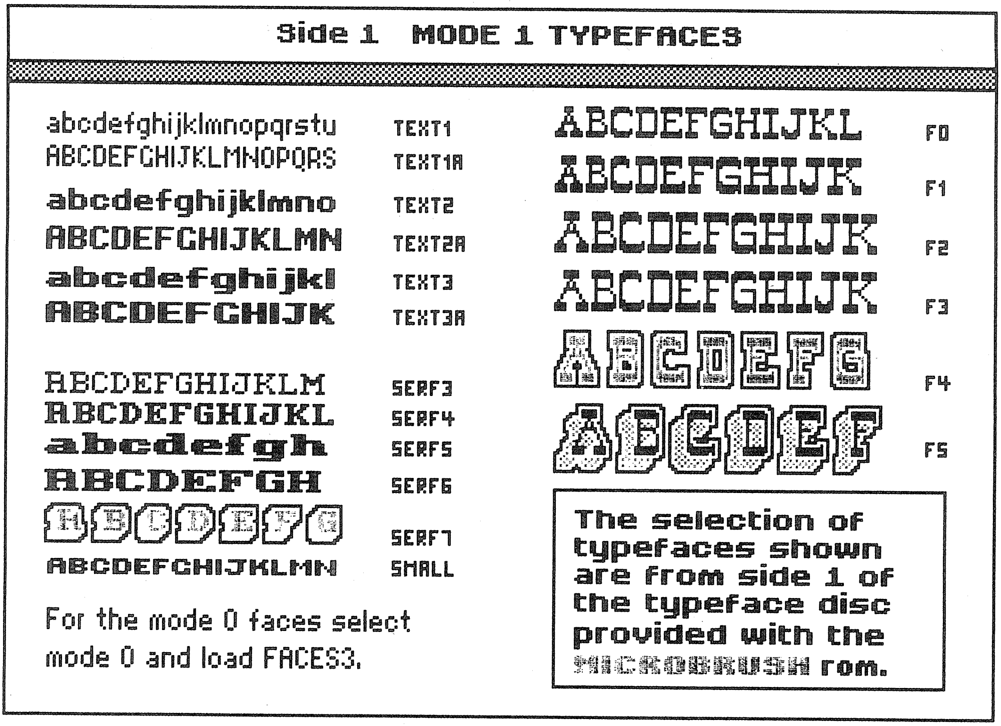
The special effects work by loading a window of the screen into its own memory, where it can be coloured, half toned, and copied back to the screen elsewhere, off-set, outlined, inverted, mirrored and even stretched and crushed by careful over-printing.

Of the fonts supplied, I used only one to produce the samples shown. Half-way through I remembered that many of the effects are already programmed into the function keys. Yes, you really can just store a word, press f2,

CONCLUSIONS

Microbrush is most distinctive in that the author genuinely uses it, and this shows in his truly impressive sample sheets. I really can't stress this enough. Graphic packages on all the micros I've used have an abysmal record of usability. And the BBC, my favourite micro, is little better. Programmers don't seem to trouble with a sensible specification, and publishers don't seem to bother even with a twenty-hour test before the software is released. After only a couple of day's use, it turns out that their limitations would leave a GCSE art student dissatisfied; routines turn out to be bugged or destroy your work without backup. Reviewing takes hours just to work out which features are usable, which are useful, which work all the time and which work only some of the time (and how to phrase it gently).

Well this program is different. It has the options the user needs, and they work, and work consistently, because the programmer uses it.



Side 1 MODE 1 TYPEFACES

abcdefghijklmnopqrstu	TEXT1	ABCDEFGHIJKL	F0
ABCDEFGHIJKLMNOPQRS	TEXT1A	ABCDEFGHIJK	F1
abcdefghijklmno	TEXT2	ABCDEFGHIJK	F2
ABCDEFGHIJKLMN	TEXT2A	ABCDEFGHIJK	F3
abcdefghijkl	TEXT3	ABCDEFG	F4
ABCDEFGHIJK	TEXT3A	ABCDEF	F5
ABCDEFGHIJKLM	SERF3		
ABCDEFGHIJKL	SERF4		
abcdefgh	SERF5		
ABCDEFGH	SERF6		
ABCDEFG	SERF7		
ABCDEFGHIJKLMN	SMALL		

For the mode 0 faces select mode 0 and load FACES3.

The selection of typefaces shown are from side 1 of the typeface disc provided with the MICROBRUSH rom.

Over the last couple of years, publishers seem to have realised that graphic packages need an 'undo' facility. Microbrush has gone beyond this. It contains many facilities to preview the effects of various functions and to reverse the effects of others. But it does not waste 20k of sideways RAM just to keep a copy of the screen. When the extra RAM is present, it uses that space for extra icons, palettes, buffers, fonts and workspace.

With a package of this size I could list many shortcomings. It would have been nicer to have the whole lot occupy only one ROM socket. It would be nice to have it use every byte of every possible RAM extension - but then I can get more out of this package than others which do so. The key- and button-pressing sequences have been well thought out, but in this sort of package they need even tighter consistency and simplicity. Colour zero is the only colour it can treat as 'background' and 'transparent'. The manual is good, but it very much needs a Beginners' Guide.

However, the range of facilities vastly exceeds my list of niggles, and the package remains the only one on the BBC which can genuinely do the job. I began by saying that I was only going to deal with part of three of the systems in the first two ROMs and their discs. The first ROM alone contains twelve such systems. So I can't pretend adequately to represent the package. The software author's consternation at this limited review is understandable! But he has been more helpful than most in making this practicable, which confirms my impression of his customer support.

Microbrush represents outstanding value for money. For a brief period a couple of years back, AB Designs may have had a competitor in AMX. Apart from that period it has always left its competitors in the shade, and probably more so now than ever.

Label Processor

John Lasruk presents an excellent utility for the simple creation of label designs, which can be printed at any size on an Epson compatible printer.

Designing and printing labels is a chore that most people dislike intensely. The usual method is to use a word processor. Unfortunately this involves having to line up headings and text correctly, and usually results in several sheets of wasted paper. This program is the answer. Labels may be designed, saved, and recalled quickly and easily. It can accommodate any continuous form, with one across label, all the way up to two inches high by four inches wide labels. It ought to work on any printer capable of standard print, six lines, and ten characters per inch. You enter your text in a white window and can then print, save or load it.

USING THE PROGRAM

When you run the program you must first enter the width and height of the label in inches. Next the program will take you into "write" mode. You can type anything you want into the white window. Use the cursor keys to move around the text if you wish. Control codes will also be accepted, but the only particularly useful ones are Ctrl-L to clear the window and Ctrl-^ to home the cursor. Ctrl-@ performs the same function as the COPY key, it takes you to the main menu.

From the menu, you may print the text in the window, save the design to disc or load a previous design and exit from the program cleanly. Be aware that loading text in from a different sized label is bound to cause formatting problems. If you use two or three different label sizes, keep their respective disc files. All label files are stored under the directory L. You should not attempt to use a directory letter as part of a file name, either for loading or saving. If you are using the program under the ADFS you will need to have created the directory L previously.

A few words about some of the more unusual features of this program. You may notice that the window can scroll down but not up. I did this deliberately, so that I can enter (say) a five line address on an eight line label, then scroll the text down to centre it. On the other hand, I didn't want to lose the top line of my text by accidentally hitting the Return key once too often. I used mode seven so that I could grab the text directly off the screen without having to buffer it or resort to trying to read each character one by one from a moving cursor position. Since the program uses byte indirection to read the screen, shadow modes and co-processors must be turned off.

PRINTING LABELS

At first, positioning labels correctly will be by trial and error. However, after a little practice this should prove to be no problem. Continuous feed labels must be installed at the extreme left of your printer. This will automatically leave a quarter inch margin on the label, which is matched on the right side. You may print as many as ten characters per inch width of label (minus four) and six lines per inch of depth (minus one). Thus a 1*3.5 inch label can hold ((10*3.5)-4)*(6-1)=155 characters.

PROGRAMMING NOTES

The printer is set up in PROCprintercodes. The actual printer control codes are contained in the DATA statements immediately following the procedure. The printer control codes terminate with a negative number.

Scrolling is inhibited by poking the value 2 into location &D0 (hint mentioned back in BEEBUG Vol.3 No.1). While it is possible to read the VDU status with a *FX command, I have yet to find a way to write the status byte legally.

PROCconvert takes certain teletext characters and converts them to ASCII for the printer. Note that for my printer, the pound sign is assigned to character 195. You may have to amend this (line 1490). Note that PROCsave also uses this convert procedure to save a label design to disc, but must convert the pound sign back again to character 96 (line 2170).

```
   10 REM Program  Label Processor
   20 REM Version  B1.02
   30 REM Author   John Lasruk
   40 REM BEEBUG   November 1988
   50 REM Program  subject to copyright
   60 :
  100 ON ERROR REPORT:PRINT" AT: ";ERL:O
SCLI"FX4":VDU3:END
  110 MODE7
  120 vpos=&70
  130 PROCassemble
  140 *FX4,2
  150 *FX11,14
  160 *FX12,5
  170 *KEY12 |H
  180 *KEY13 |I
  190 *KEY14 |J
  200 *KEY15 |K
  210 *KEY11 |@
  220 PROClabelsize
  230 labels=0
  240 PROCsetup
  250 VDU15
  260 Q=&7C02+(15-D*6)*40
  270 REPEAT
  280 CALL&900
  290 PROCprintit
  300 UNTIL FALSE
  310 END
  320 :
 1000 DEFPROCsetup
 1010 CLS
 1020 FOR X=0 TO 1:VDU132,157,141
 1030 PRINT SPC(6);"BEEBUG LABEL PROCESS
OR"
 1040 VDU31,0,15
 1050 PRINT SPC(4);"USE CURSOR KEYS TO E
DIT LABEL AND"
 1060 PRINT SPC(6);"COPY KEY FOR LABEL M
ENU"'
 1070 VDU31,0,(14-D*6)
 1080 FOR Y=0 TO (D*6)
 1090 VDU157,132:PRINT SPC(W+2);
 1100 IF W<36:VDU156,13,10
 1110 NEXT
 1120 PROCwindow1
 1130 CLS
 1140 ENDPROC
 1150 :
 1160 DEFPROCwindow1
 1170 VDU28,2,13,(W+1),(15-D*6),30
 1180 ENDPROC
 1190 :
 1200 DEFPROCwindow2
 1210 VDU28,0,24,39,17
 1220 CLS
 1230 ENDPROC
 1240 :
 1250 DEFPROCexit
 1260 *FX4
 1270 CLS
 1280 ENDPROC
 1290 :
 1300 DEFPROCprintit
 1310 PROChowmany
 1320 IF labels=0:PROCwindow1:ENDPROC
 1330 VDU2
 1340 PROCprintercodes
 1350 FOR X=1 TO labels
 1360 FOR line= Q TO Q+(40*(D*6-1)) STEP
 40
 1370 FOR character=line TO line+W-1
 1380 letter=?character: PROCconvert
 1390 VDU1,letter
 1400 NEXT:VDU1,13:NEXT:NEXT
 1410 VDU3
 1420 CLS:PRINT''" CONTINUE?"
 1430 KEY=GET:CLS
 1440 IFKEY<>ASC"N":IF KEY<>ASC"n":PROCw
indow1:ENDPROC
 1450 VDU22,7:PROCexit
 1460 END
 1470 :
 1480 DEFPROCconvert
 1490 IF letter=35:letter=195
 1500 IF letter=95:letter=35
 1510 IF letter=96:letter=95
 1520 ENDPROC
 1530 :
 1540 DEFPROChowmany
 1550 labels=0
 1560 PROCwindow2
 1570 PROCmenuprint
 1580 IF INSTR("SLEPslep",KEY$)=0:CLS:EN
DPROC
 1590 IF KEY$="S"ORKEY$="s":PROCsave:CLS
:ENDPROC
 1600 IF KEY$="L"ORKEY$="l":PROCload
 1610 IF KEY$="E"ORKEY$="e":PROCexit:VDU
22,7:END
 1620 CLS
 1630 INPUT''SPC(3);"HOW MANY LABELS DO
YOU WISH TO PRINT? "labels
 1640 CLS
 1650 ENDPROC
 1660 :
```

```
 1670 DEFPROCprintercodes
 1680 RESTORE
 1690 REPEAT
 1700 READ code%
 1710 IF code%>=0:VDU1,code%
 1720 UNTIL code%<0
 1730 ENDPROC
 1740 DATA 27,71,27,69,-99
 1750 :
 1760 DEFPROClabelsize
 1770 CLS
 1780 FORX=0TO1
 1790 VDU131,157,129,141
 1800 PRINT SPC(4);"INDICATE SIZE OF LAB
EL:"
 1810 NEXT
 1820 VDU10,10:FOR X=1 TO 2:VDU141
 1830 PRINT" The width of your labels in
 inches: "
 1840 NEXT
 1850 INPUT W: W=(W*10)-4
 1860 VDU10,10:FOR X=1 TO 2:VDU141
 1870 PRINT" The height of your labels i
n inches:"
 1880 NEXT
 1890 PRINT SPC(3);"(The distance from t
he top of one"
 1900 PRINT SPC(3);"to the top of the ne
xt.)"
 1910 INPUT D
 1920 ?vpos=D*6-2
 1930 ENDPROC
 1940 :
 1950 DEFPROCmenuprint
 1960 FOR X=0 TO 1
 1970 VDU132,157,131,141
 1980 PRINT"   PLEASE SELECT ONE OF THES
E:"
 1990 NEXT
 2000 PRINT'SPC(3);"P";SPC(3);"PRINT THE
 LABEL IN THE WINDOW"
 2010 PRINT SPC(3);"S";SPC(3);"SAVE THE
LABEL TO DISC"
 2020 PRINT SPC(3);"L";SPC(3);"LOAD A LA
BEL FROM DISC"
 2030 PRINT SPC(3);"E";SPC(3);"END THIS
PROGRAM"
 2040 VDU132,157
 2050 KEY$=GET$
 2060 ENDPROC
 2070 :
 2080 DEFPROCsave
 2090 CLS
 2100 INPUT''" ENTER A NAME FOR THIS FIL
E: "name$
 2110 IF LENname$>7:VDU7:PRINT"TOO LONG!
":GOTO2100
 2120 VDU21
 2130 OSCLI"SPOOL L."+name$
 2140 FOR line= Q TO Q+(40*(D*6-2)) STEP
 40
 2150 FOR character=line TO line+W-1
 2160 letter=?character: PROCconvert
 2170 IF letter=195:letter=96
 2180 VDUletter
 2190 NEXT:NEXT
 2200 *SPOOL
 2210 VDU6:CLS
 2220 ENDPROC
 2230 :
 2240 DEFPROCload
 2250 CLS
 2260 INPUT''" ENTER THE LABEL FILE NAME
 (or *command) "disc$
 2270 IF INSTR(disc$,"*")=1:OSCLI disc$:
PRINT" >>SPACE<<";GET:GOTO2250
 2280 PROCwindow1:?&D0=2:VDU30
 2290 OSCLI"TYPE L."+disc$
 2300 ?&D0=8:PROCwindow2:CLS
 2310 ENDPROC
 2320 :
 2330 DEFPROCassemble
 2340 FOR X=0 TO 2 STEP2
 2350 P%=&900
 2360 [
 2370 OPT X
 2380 .read
 2390 JSR&FFE0:CMP#13:BEQ return
 2400 CMP#0: BEQ getlost
 2410 .test
 2420 PHA:LDA#134:JSR&FFF4
 2430 CPYvpos: BEQinhibit
 2440 PLA
 2450 .write
 2460 JSR&FFEE:JMPread
 2470 .getlost RTS
 2480 .return
 2490 LDA#2:STA&D0
 2500 JSR&FFE7:LDA#8:STA&D0
 2510 JMPread
 2520 .inhibit
 2530 LDA#2:STA&D0:PLA:JSR&FFEE
 2540 LDA#8:STA&D0:JMPread
 2550 ]
 2560 NEXT
 2570 ENDPROC
```

Formation Snatching

David Spencer takes a look at a variation on the theme of printer dumps.

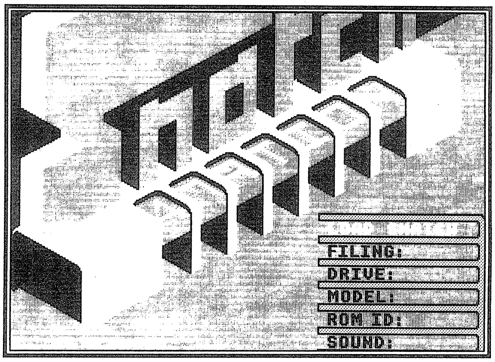
Product	**Snatch**
Supplier	**4Mation Educational Resources Linden Lea, Rock Park, Barnstaple, Devon EX32 9AQ. Tel. (0271) 45566**
Price	**£18.40 inc. VAT**

In this day and age, it must be a very bold (or foolish) company that launches a new printer dump for the Beeb. However, that is just what 4Mation has done with its *Snatch* package. *Snatch* is in fact more than a printer dump - it is also a 'screen grabber' that allows pictures to be saved to disc while a program is running. As an added bonus a complete pixel editor is included.

Snatch comes in the form of a very colourful user guide with the software on 5.25" disc held in a flap on the inside back cover. To add to the colour the disc is white with the title and 4Mation's logo screen printed on it. The disc sleeve is transparent plastic rather than the usual white paper. The disc is in fact a flippy, although both sides contain the software in 40 track DFS format. It would have been much more useful to have one side 40 track, and the other 80.

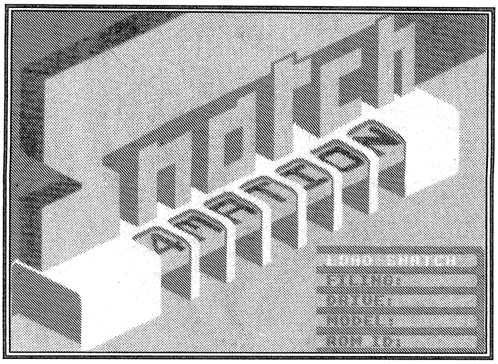
Although *Snatch* comes on disc it is actually a sideways ROM image that has to be loaded into sideways RAM. A loader program is provided for this purpose, and this is invoked when the *Snatch* disc is started with Shift-Break. The loader allows you to specify which bank of sideways RAM is to be used, and also other options such as the filing system that will be used to save grabbed screens. If you are not sure about which banks contain sideways RAM, then the disc contains a ROM lister program that prints the names of the ROMs in each slot. Sideways RAM is printed in a different colour and can easily be identified. Once loaded, *Snatch* is initialised by pressing Ctrl-Break.

Before using *Snatch* to grab screens, it is necessary to create a picture disc. This is done by formatting the disc to the format specified when *Snatch* was loaded and then typing *SNATCH. This will bring up a two-option menu, the first of which is 'Prepare Disc'. Selecting this option will create a set of blank screens on the new picture disc, together with information about those screens. *Snatch* treats graphics and teletext screens separately, and the number that can be fitted on one disc depends on the format. All discs can hold fifteen teletext screens, but the number of graphics screens ranges from four on a 40 track DFS disc to thirty-one on a double sided 80 track ADFS disc.



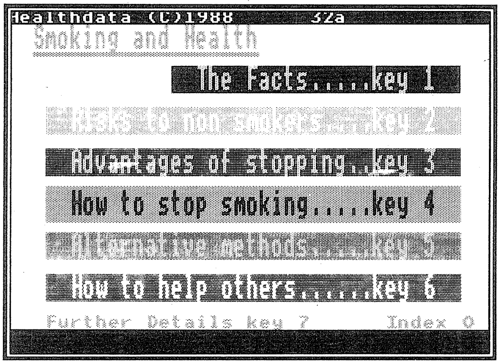
Epson Mode 1 Screen Dump

Using *Snatch* to grab a screen from within a running program is simplicity itself. All you need to do is ensure that a picture disc is in the correct drive and press Shift, Ctrl and X together. The screen will be frozen and saved to the disc. Once the screen has been saved pressing Space will continue execution of the program. My only quibble here is that the screen is automatically saved. With fast-changing displays it would be nice to freeze the screen and then look at the result before saving it. If necessary you could continue execution without saving the screen.



Integrex Mode 1 Screen Dump

Displaying one or more grabbed screens is also simple. Typing *SNATCH and selecting the 'Display Screens' option brings up a control screen. From this screen you can find out how many pictures are stored on the disc and display them. There is also an option to wipe one or all of the pictures. This can be reversed with the Restore feature. An Enlarge option lets any area of a picture be enlarged to full screen size, and this can be re-saved if desired.



Healthdata (C)1988 32a
Smoking and Health
The Facts.....key 1
Risks to non smokers....key 2
Advantages of stopping..key 3
How to stop smoking.....key 4
Alternative methods.....key 5
How to help others......key 6
Further Details key 7 Index 0

Epson Mode 7 Screen Dump

Dumping screens to a printer using *Snatch* is as easy as saving them to disc. At any time pressing Shift, Ctrl and E together will dump to an Epson compatible printer, while Shift, Ctrl and I will perform the dump using an Integrex 132 Colourjet. Both printer dump routines allow either small or large dumps to be printed, and the size to use is chosen by pressing 'L' or 'S' after the dump is invoked. Due to the limitations of some printers, dumps in modes 0 and 7 may only be made in the large form on Epson compatible printers. A further option allows the use of normal or high-density modes when using an Integrex 132. High-density mode gives a better result but is slower.

As a idea of the speed of the printer dumps, the following table gives the timing (in minutes) taken to perform large dumps of various modes on various printers. These figures are taken from the *Snatch* manual.

LARGE		mode	
	0	1	7
Epson LQ800	3.5	3.25	2
Epson LX80	4	3.25	2.5
Integrex	14.25	13.25	2
SMALL		mode	
	0	1	7
Epson LQ800	-	3.25	-
Epson LX80		-2.25	-
Integrex	3.5	3.25	0.75

DOCUMENTATION

The user guide for *Snatch* appears to be very professional. The twenty-two pages are printed in full colour on good quality paper with colourful hi-gloss covers. The manual is very chatty in places, and while it fully explains everything that the average user will need, I found it a little patronising in places. Not all the explanations are as clear as they could be. In particular, I had problems following the section that explains setting up a picture disc, and trial and error proved to be the only solution.

CONCLUSION

Are 4Mation bold or foolish? I think the answer must be bold. *Snatch* is a product that you either need or you don't. Many people who use their printers simply for letters and listings probably won't need a screen freezer and dumper. On the other hand, I am sure there are lots of people who will find *Snatch* the best thing since sliced bread. It is up to you to decide if you need *Snatch*, but I can say that it does the job it is designed to, and it does it well. The price of £18.40 is also reasonable for a product of this nature. Overall I liked *Snatch*. B

Graphic Design With ASTAAD 3

In this, the second instalment of our ASTAAD design package, David Demaine adds a host of new graphics features.

Last month we published the basic building blocks from which to create a very powerful design package. This month's listing provides all of the remaining graphics options. The keystrip is reprinted in this month's issue, with all the working functions highlighted in the same manner as last month.

It is most important to type in this month's listing with absolute adherence to the line numbers. Do not worry if the listing appears to overwrite parts of the previous listing, as we have incorporated suggestions that have been made by members since last month. Also, make sure that you do not renumber the program: if you do, it will be practically impossible to enter next month's final listing.

THE NEW FEATURES

The remaining functions, published this month, are all very short and simple, but make maximum use of the graphics capabilities of the Master series.

These features fall into two classes, graphics functions which initiate a graphics operation, and toggle functions which affect the operation of the primary operations including some of those described last month.

NEW GRAPHICS FUNCTIONS

Ctrl-f6 allows you to set your own scale to work with. There are three options. When selected, you will be asked "Simple scale?". This is a simple multiplier of the graphics screen (the default setting is 1). To move to the next option, just enter 0.

The second and third options enable you to specify the coefficient for any printer of your choice. The coefficient required for a Canon PW-1080A printer and the BEEBUG screen dump ROM, Dumpmaster, is contained in line 1600 of the program, and may have to be adjusted for other dump programs and printers. You may specify a printed scale in *units/cm* or with the third option *units/inch*. The units referred to are those displayed as the x and y co-ordinates, and the vector diagonal, of the cursor position. The units may represent any magnitude you please, miles, millimetres or microns.

The new code includes a more complicated shape function - *Polygon*. This enables you to draw polygons with any number of sides, and also allows the definition of isometric tilt and rotation. When function key f4 is pressed, you will be asked for the maximum diameter over the points, and then the number of sides. If you enter less than 3 or more than 180, the function is aborted. You are then given the option to define more details (answer Y or N to the prompt). If you select this option, you are first asked for the base angle. This is the angle, measured anti-clockwise, between the horizontal and the base of the polygon (default 0). This allows any polygon to be rotated as required about its centre.

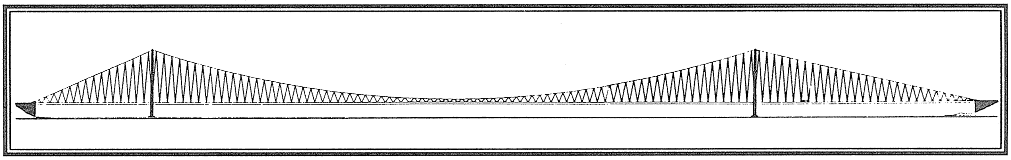
The other two pieces of data are related, and allow any polygon to be tilted (rotated) about any axis in its plane, and the resulting image displayed as it would appear projected onto the two dimensional screen. The tilt axis always passes through the centre of the polygon. Thus, for example, a tilt axis of 90°, and a tilt angle of 45°, rotates the polygon about a vertical axis. As this happens, the polygon appears to get thinner. Tilting with tilt axis zero, will tilt the polygon about a horizontal axis. Tilting by 90° about any axis results in a straight line, as the polygon is then being viewed end-on. If this sounds complicated, I suggest you try it experimentally, but only varying one or two parameters at a time.

An ellipse drawing function is also included, and again the ellipse may be defined with its axes at any angle.

NEW TOGGLE FUNCTIONS

The remainder of the new features are *toggles* which affect the operation of the various graphics functions. Some of the toggles provide two options, like the *Infill/Outline* and *Colour/Reverse*. However, one is a *multi-toggle* which enables a choice of several options. The multi-toggle always selects options in the same order, starting at the first with the initial key press. Consecutive key presses must be at less than half second intervals.

Shift-f1 toggles between copy or move (for areas of the screen). Shift-f7 causes the cursor to either transfer to the end of a line when the *Line* function is used, or remain in the same position. Shift-f8 is a three option multi-toggle. It always gives the accelerating cursor mode when first pressed, but a second quick press gives cursor steps of 16 standard graphics pixels, useful for direct screen text editing, while a third quick press gives a mode in which the cursor only moves in single displayed graphics pixels, 2 horizontally and 4 vertically, useful for detailed work. Shift-f9 toggles between a displayed graphics origin, which is reset to the current cursor position, or the standard screen origin in the bottom left hand corner. The Tab function is re-initialised when this toggle is used, to avoid ambiguity.



Ctrl-f2 is a multi-toggle which gives the choice of solid or dotted lines to be used by the *Line* function. This also applies when the Copy key is used to draw a line or a box to the last Tab position. Five different dot, dash, and dot-dash options are provided in the program. When in operation, they are represented by their hex code in the control bar at the top of the screen. They may be changed as you wish, (see DATA statement in line 4760) but the last value must be zero.

The new two-way toggle (Ctrl-f1) allows you to switch between drawing a line or a rectangle, from the cursor to the last Tab position. This is very handy for drawing boxes and grids. Ctrl-f3 releases the cursor to travel off the graphics screen, which is useful for constructional purposes. Its use is illustrated by this month's feature illustration, the drawing of the Humber Bridge, where the centre of the bridge is the geometrical centre of reference, and is removed many screen widths away from most of the screens. The very high resolution shows what can be achieved when a drawing is prepared on several screens, which are then reduced and joined together in a montage.

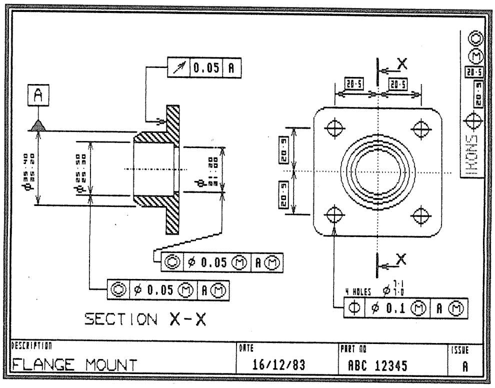
THE SHAPE FUNCTIONS

The keys f2, f3 and f4 are associated with Shift-f2, f3 and f4, and between them provide six standard shape functions, line or arrow, circle or arc, and polygon or ellipse. The coefficients defining each of these shapes are stored until they are redefined, and may be recalled for plotting with the Repeat f2, f3, f4 key (key f5). For example, if you have been using the line function, but wish to draw a previously defined circle, simply press Shift-f3 until the header message says "Toggling to circle" and then press f5.

The shapes provided are quite simple, but can easily be redefined to be anything you wish. In fact, the *Circle* and the *Ellipse* shapes can be drawn using the *Polygon* shape with a large number of sides, and for an ellipse by tilting about an axis. Tilting any polygon through 90° produces a straight line. Thus any of the specific circle, line or ellipse functions could be replaced by others of your own choice in the program, and the polygon feature used to reproduce these when needed.

The parameters for redefining any of the shapes are requested by simple messages in the heading. A value greater than 360 degrees for the last requested angle aborts the entry, and retains the last defined parameters except for the circle (for which data entry can be aborted by a zero radius).

Finally two further features: Escape causes the cursor to return to the screen centre, while Shift-Escape clears the screen completely.



Next month we conclude this short series with the addition of a flexible text writing facility for displaying any size text at any angle, anywhere on the drawing, the feature from which ASTAAD gets its name.

```
   10 REM Program  Astaad (Part 2)
   20 REM Version  B3.03
   30 REM Author   David Demaine
   40 REM Based On Original By Tim Tonge
   50 REM BEEBUG   November 1988
   60 REM Program  subject to copyright
   70 :
  100 IF J%=&7FFFFF chain%=TRUE ELSE cha
in%=FALSE
  110 J%=13:DIM OS% 160
  130 IF NOT chain% MODE 128
  150 PROCsetup:quit%=FALSE:ON ERROR PRO
Cerror
  800 DEF PROCerror
  810 IF ERR=17 AND NOT INKEY-1 ENDPROC
  820 VDU4,31,6,0
  830 REPORT:PRINT " at line "; ERL;" ha
s occurred;"'"Continue? (Y/N)";
  840 IF quit% GOTO 860
  850 IF (GET AND &DF)=ASC"Y" VDU 12,5:P
ROCrehead:ENDPROC
  860 *FX4,0
  870 *FX229,0
  880 VDU22,128:REPORT:PRINT " at line "
; ERL'
  890 @%=&90A:ON ERROR OFF
  900 *DELETE NEERCS
  910 END
  920 :
 1210 PROCrehead:ENDPROC
 1255 IF NOT chain% CLG
 1285 *FX229,1
```

```
 1444 arrowlen=400:arrowang=5*PI/6:del=R
AD(16)
 1446 arcrad=450:arcstart=-4*PI/6:arcsub
t=4*PI/6
 1455 semimaj=400:semimin=400*COS(PI/6):
ellipsang=PI/3
 1600 scale$=STRING$(3,CHR$32):scl$="tim
es":scale=1.0:scalep=1.0:scaler=56.705
 1670 @%=&00020108:PRINTTAB(0,0)fn$" "cu
rs$" "soft$" "copy$" "dot$" "marg$" "fil
l$" "ecf$" "col$" "fix$" "accel$" "orig$
" x=" (x-x4)*scale" y=" (y-y4)*scale;
 1690 @%=&00020108:PRINT scl$ scale$" "
vector$ vector*scale;
 2030 ON J% PROCnoproc,PROCareamove,PROC
line,PROCcircle,PROCpoly,PROCrepeat,PROC
move,PROCdraw,PROCrubout,PROCdelete,PROC
none,PROCcopy ELSE ENDPROC
 2070 ON J%-16 PROCnoproc,PROCcopymove,P
ROClinearrow,PROCcircarc,PROCpolylipse,P
ROCfill,PROCcolour,PROClinetog,PROCspeed
,PROCorigin ELSE ENDPROC
 2110 ON J%-32 PROCdump,PROCvector,PROCs
olidot,PROCmargins,PROCnoproc,PROCnoproc
,PROCscale,PROCnoproc,PROCsave,PROCload
ELSE ENDPROC
 2490 :
 2510 IF repeat% PROCmessage("Repeating
line") ELSE p1=FNinput("Length?")
 2620 :
 3190 @%=&00020101:PRINT '"Scale factor
is "; scale '"Printed scale is "' SPC(10
); scale*scaler;" units to one cm"
 3440 DEF PROCline:IF J%<>3 ENDPROC
 3450 lastJ%=J%:IF arrow% PROCarrow ELSE
 PROCliner
 3460 ENDPROC
 3470 :
 3480 DEF PROCarrow:fn$=arrow$
 3490 IF repeat% PROCmessage("Repeating
arrow") ELSE p1=FNinput("Length?")
 3500 IF NOT repeat% p2=FNinput("Angle (
deg)?"):IF ABS(p2)>360.9 PROCrehead:ENDP
ROC
 3510 IF NOT repeat% arrowlen=p1/scale:a
rrowang=RAD(p2)
 3520 MOVE x,y:PLOT linecode%+dot%,x+arr
owlen*COS(arrowang),y+arrowlen*SIN(arrow
ang)
 3530 MOVE x,y:ang=arrowang+del:PLOT lin
ecode%,x+24*COS(ang),y+24*SIN(ang):MOVE
x,y:ang=arrowang-del:PLOT linecode%,x+24
*COS(ang),y+24*SIN(ang)
 3540 IF repeat% PROCdelay
 3550 PROCrehead:ENDPROC
 3560 :
 3570 DEF PROCcircle:IF J%<>4 ENDPROC
 3580 lastJ%=J%:IF circ% PROCcirc ELSE P
ROCarc
 3590 ENDPROC
 3600 :
 3610 DEF PROCarc:fn$=circ$
 3620 IF repeat% PROCmessage("Repeating
arc"):PROCheading ELSE p1=FNinput("Radiu
s of arc?")
 3630 IF NOT repeat% p2=FNinput("Start a
ngle of arc?")
 3640 IF NOT repeat% p3=FNinput("Subtend
ed angle of arc?"):IF ABS(p3)>360.9 PROC
rehead:ENDPROC
 3650 IF NOT repeat% arcrad=p1/scale:arc
start=RAD(p2):arcsubt=RAD(p3)
 3660 MOVE x+arcrad*COS(arcstart),y+arcr
ad*SIN(arcstart)
 3670 PLOT shapcode%+160,x+arcrad*COS(ar
cstart+arcsubt),y+arcrad*SIN(arcstart+ar
csubt)
 3680 IF repeat% PROCdelay
 3690 PROCrehead:ENDPROC
 3700 :
 3710 DEF PROCpoly:IF J%<>5 ENDPROC
 3720 lastJ%=J%:IF poly% PROCpolygon ELS
E PROCellipse
 3730 ENDPROC
 3740 :
 3750 DEF PROCpolygon:fn$=poly$
 3760 IF repeat% PROCmessage("Repeating
polygon") ELSE p1=FNinput("Max diam over
 points?")
 3770 IF NOT repeat% p2%=FNinput("No. of
 sides?"):IF p2%<3 OR p2%>180 PROCrehead
:ENDPROC
 3780 IF NOT repeat% ans$=FNcheck("Furth
er definition?")
 3790 IF NOT repeat% IF ans$="Y" OR ans$
="y" PROCtilt
 3800 IF NOT repeat% IF ans$<>"Y" AND an
s$<>"y" p3=0:p4=0:p5=0
```

```
 3810 IF NOT repeat% IF ABS(p5)>360.9 PR
OCrehead:ENDPROC
 3820 IF NOT repeat% polyrad=p1/2/scale:
polysides%=p2%:polybase=RAD(p3):polyrot=
RAD(p4):polytilt=RAD(p5)
 3830 cosgamma=COS(polyrot):singamma=SIN
(polyrot):costilt=COS(polytilt)
 3840 alpha=2*PI/polysides%
 3850 betaf=(alpha-PI)/2+polybase
 3860 cosbeta=COS(betaf):sinbeta=SIN(bet
af)
 3870 cosdiff=cosbeta*cosgamma+sinbeta*s
ingamma
 3880 sindiff=(sinbeta*cosgamma-cosbeta*
singamma)*costilt
 3890 lastx=FNpx:lasty=FNpy
 3900 FOR I%=1 TO polysides%
 3910 beta=I%*alpha+betaf
 3920 cosbeta=COS(beta):sinbeta=SIN(beta
)
 3930 cosdiff=cosbeta*cosgamma+sinbeta*s
ingamma
 3940 sindiff=(sinbeta*cosgamma-cosbeta*
singamma)*costilt
 3950 MOVE x,y:MOVE lastx,lasty:lastx=FN
px:lasty=FNpy
 3960 PLOT shapcode%+fill%*10,lastx,last
y
 3970 NEXT I%
 3980 IF repeat% PROCdelay
 3990 PROCrehead:ENDPROC
 4000 :
 4010 DEF PROCtilt
 4020 p3=FNinput("Base angle?")
 4030 p4=FNinput("Tilt axis?")
 4040 p5=FNinput("Tilt angle?")
 4050 ENDPROC
 4060 :
 4070 DEF FNpx:=x+polyrad*(cosdiff*cosga
mma-sindiff*singamma)
 4080 :
 4090 DEF FNpy:=y+polyrad*(sindiff*cosga
mma+cosdiff*singamma)
 4100 :
 4110 DEF PROCellipse:fn$=circ$
 4120 IF repeat% PROCmessage("Repeating
ellipse") ELSE p1=FNinput("Major diamete
r of ellipse?")
 4130 IF NOT repeat% p2=FNinput("Minor d
iameter of ellipse?")
 4140 IF NOT repeat% p3=FNinput("Angle o
f major axis?"):IF ABS(p3)>360.9 PROCreh
ead:ENDPROC
 4150 IF NOT repeat% semimaj=p1/2/scale:
semimin=p2/2/scale:ellipsang=RAD(p3)
 4160 MOVE x,y:PROCplotellipse(x,y,semim
aj,semimin,ellipsang)
 4170 IF repeat% PROCdelay
 4180 PROCrehead:ENDPROC
 4190 :
 4200 DEF PROCplotellipse(centrex,centre
y,semimajor,semiminor,angle)
 4210 LOCAL cosang,sinang,maxy,shearx,sl
icewidth
 4220 cosang=COS(angle):sinang=SIN(angle
)
 4230 maxy=SQR((semiminor*cosang)*(semim
inor*cosang)+(semimajor*sinang)*(semimaj
or*sinang))
 4240 shearx=(semimajor*semimajor-semimi
nor*semiminor)*cosang*sinang/maxy
 4250 slicewidth=semimajor*semiminor/max
y
 4260 MOVE centrex,centrey:PLOT0,slicewi
dth,0
 4270 PLOT shapcode%+fill%+192,centrex+s
hearx,centrey+maxy
 4280 ENDPROC
 4290 :
 4300 DEF PROCrepeat
 4310 IF J%<>6 ENDPROC
 4320 repeat%=TRUE
 4330 J%=lastJ%
 4340 ON lastJ%-2 PROCline,PROCcircle,PR
OCpoly
 4350 repeat%=FALSE:ENDPROC
 4360 :
 4370 DEF PROCcopymove:IF J%<>18 ENDPROC
 4380 IF copy% copy%=FALSE:copy$="move":
areacode%=0 ELSE copy%=TRUE:copy$="copy"
:areacode%=2
 4390 PROCheading:ENDPROC
 4400 :
 4410 DEF PROClinearrow:IF J%<>19 ENDPRO
C
 4420 lastJ%=3:IF arrow%=TRUE arrow%=FAL
SE:arrow$="Line  " ELSE arrow%=TRUE:arro
w$="Arrow "
 4430 PROCmessage("Toggling to "+arrow$)
:PROCdelay:PROCrehead:ENDPROC
 4440 :
 4450 DEF PROCcircarc:IF J%<>20 ENDPROC
 4460 lastJ%=4:IF circ% circ%=FALSE:circ
$="Arc   " ELSE circ%=TRUE:circ$="Circle
"
 4470 PROCmessage("Toggling to "+circ$):
PROCdelay:PROCrehead:ENDPROC
 4480 :
 4490 DEF PROCpolylipse:IF J%<>21 ENDPRO
C
 4500 lastJ%=5:IF poly% poly%=FALSE:poly
$="Ellips" ELSE poly%=TRUE:poly$="Polygn
"
 4510 PROCmessage("Toggling to "+poly$):
PROCdelay:PROCrehead:ENDPROC
```

```
 4520 :
 4530 DEF PROClinetog:IF J%<>24 ENDPROC
 4540 IF fix% THEN fix%=FALSE:fix$="Tran
" ELSE fix%=TRUE:fix$="Fix "
 4550 PROCheading:ENDPROC
 4560 :
 4570 DEF PROCspeed:IF J%<>25 ENDPROC
 4580 IF TIME>t% accel%=TRUE:s%=4:accel$
="Accl":PROCheading:TIME=0:ENDPROC
 4590 accel%=FALSE:IF s%=4 s%=16:accel$=
" 16 ":PROCheading:TIME=0:ENDPROC
 4600 s%=4:accel$="2,4 ":PROCheading:TIM
E=t%+1:ENDPROC
 4610 :
 4620 DEF PROCorigin:IF J%<>26 ENDPROC
 4630 IF x4=0 AND y4=0 THEN x4=x:y4=y:or
ig$="Rel " ELSE x4=0:y4=0:orig$="Abs "
 4640 x1=x:y1=y:J%=13:PROCheading:ENDPRO
C
 4650 :
 4660 DEF PROCvector:IF J%<>34 ENDPROC
 4670 IF vector% vector%=FALSE:vector$="
Diagonal" ELSE vector%=TRUE:vector$=" Ve
ctor:"
 4680 PROCheading:copycode%=5:ENDPROC
 4690 :
 4700 DEF PROCsolidot:IF J%<>35 ENDPROC
 4710 lastJ%=3
 4720 IF TIME>t% dot%=0:dot$="Line":PROC
heading:RESTORE 4760:TIME=0:ENDPROC ELSE
 dot%=16
 4730 READ hex%:dot$="&"+STR$~hex%+" "
 4740 IF hex%=0 dot%=0:dot$="Line":TIME=
t%+1 ELSE VDU 4,23,6,hex%|5:TIME=0
 4750 PROCheading:ENDPROC
 4760 DATA &AA,&EE,&6F,&27,&88,0
 4770 :
 4780 DEF PROCmargins:IF J%<>36 ENDPROC
 4790 IF marg% marg%=FALSE:marg$="Edge"
ELSE marg%=TRUE:marg$="MarR"
 4800 PROCheading:ENDPROC
 4810 :
 4820 DEF PROCscale:IF J%<>39 ENDPROC
 4830 fn$="Scale ":s=FNinput("Simple sca
le")
 4840 IF s<>0 scale=1.0*s:scalep=scale:s
cl$="times":scale$="   ":PROCrehead:ENDP
ROC
 4850 s=FNinput("Printed units / cm")
 4860 IF s<>0 scale=s/scaler:scalep=s:sc
l$="units":scale$="/cm":PROCrehead:ENDPR
OC
 4870 s=FNinput("Printed units / inch")
 4880 IF s<>0 scale=s/scaler/2.54:scalep
=s:scl$="units":scale$="/in":PROCrehead:
ENDPROC
 4890 PROCrehead:ENDPROC
```

Basic Program Resequencer

Put your programs into order with the latest Basic utility from David Spencer.

How often have you wanted to move lines of a Basic program around? Perhaps you needed to take some lines from the middle of a program and put them at the end to form a procedure. Or you might want to put all the procedures and functions into alphabetical order. Or how about duplicating a set of lines that are needed more than once in the program. The *BEEBUG Program Resequencer* allows you to perform any of these operations. In essence, this utility will either move or copy a particular group of lines to any place within the program. Additionally, the program is renumbered as necessary to ensure that the line numbers remain in order.

The *BEEBUG Program Resequencer* is in the form of a sideways ROM image that must be loaded into sideways RAM. You must therefore have at least one bank of sideways RAM installed in your machine. To enter the *BEEBUG Program Resequencer* type in the listing given below. The listing should then be saved and the program run. This will assemble the ROM image and save it under the name 'RESEobj'. The method of loading the ROM image depends on the sideways RAM that is installed. For a B+128, a Master or a Compact, the command:

```
    *SRLOAD RESEobj 8000 WQ
```

will do the job. For a model B with sideways RAM fitted, the RAM user guide should give loading instructions. Once the image is loaded, press Ctrl-Break to initialise it.

The *BEEBUG Program Resequencer* provides two star commands. These are:

`*BRCOPY` to copy a group of program lines and

`*BRMOVE` to move a group of program lines

The first of these just replicates the lines, while the second is equivalent to copying the lines then deleting the original. Both commands take up to three parameters, separated by either spaces or commas, for example:

```
    *BRCOPY [<start>] [<end>] [<destination>]
```

<start> is the line number of the first line to copy or move, and <end> is the number of the last line in the block. <destination> is the number of the line after which to place the copied block. For example:

```
    *BRMOVE 100 200 1000
```

would move lines 100-200 to immediately after line 1000. An error will be given if the destination lies within the block being moved.

All three parameters are treated in the same way as the LIST command does. So for example, if a program was numbered in increments of 10 then the command:

```
    *BRMOVE 91 209 1009
```

would perform the same function as:

```
    *BRMOVE 100 200 1000
```

One of two of the parameters may be omitted, in which case default values will be assumed. If the start line is omitted then the first line of the program is used, and if the end line is not present then the last line of the program is assumed. If no destination is given then the block will be copied or moved to the end of the program. If either the start or end line numbers are omitted then commas must be used to show that the particular number is not present. Because the destination is the last parameter this can simply be omitted altogether. For example, to move all the lines after 10000 to a point after line 3000 you could use:

```
    *BRMOVE 10000,,3000
```

The absence of a number between the two commas shows that no end line is given. Similarly:

```
    *BRCOPY 100,1000
```

will copy all lines between 100 and 1000 to the end of the program. Finally:

```
    *BRMOVE ,500
```

would move all the lines up to and including 500 to the end of the program. The leading comma shows that no start line number is given.

Obviously after moving or copying a block of program the line numbers can be out of order. The *BEEBUG Program Resequencer* resolves this by renumbering the new lines, and all those following them in increments of ten starting at the destination line number. For example, the command:

```
*BRMOVE 100 200 1000
```

would perform the move and then number the moved lines from 1010 up to 1110, with the lines that followed 1000 being numbered from 1120 upwards. This ensures that the line numbers always increase from one to another, although Basic will execute a program that has out of sequence lines.

There is one possible area of confusion when lines are renumbered after *BRCOPY has been used. If a block of lines is copied to a point in a program before its current position, and a line within that block is referred to elsewhere in the program, then the program will be renumbered so that all the references refer to the copy of the appropriate line in the new block.

```
   10 REM Program Program Resequencer
   20 REM Version B1.0
   30 REM Author  David Spencer
   40 REM BEEBUG  November 1988
   50 REM Program subject to copyright
   60 :
  100 page=&18:ramtop=&C000
  110 min=&70:max=&72:lin=&74
  120 lptr=&76:temp=&78:start=&7A
  130 end=&7C:dest=&7E:curl=&80
  140 bs=&82:bd=&84:bl=&86
  150 ytemp=&88:type=&89:pend=&8A
  160 him=&8C:bline=&8E:size=&60
  170 btop=&62
  180 osnewl=&FFE7:oswrch=&FFEE
  190 osbyte=&FFF4
  200 DIM code 3000
  210 FORpass=4 TO 7 STEP3
  220 P%=&8000:O%=code
  230 [OPT pass
  240 EQUB 0:EQUB 0:EQUB 0
  250 JMP service
  260 EQUB &82:EQUB cw AND &FF:EQUB 1
  270 EQUS "BEEBUG Program Resquencer"
  280 .cw EQUB 0
  290 EQUS "(C) BEEBUG 1988":EQUB 0
  300 .service CMP #4:BEQ comm:RTS
  310 .comm TYA:PHA:LDX #0
  320 .comm2 LDA (&F2),Y:AND #&DF
  330 CMP ctab,X:BNE comm4:INY:INX
  340 LDA ctab,X:BPL comm2:LDA (&F2),Y
  350 CMP #&21:BCS nextc
  360 .comm3 JSR sskp:LDA ctab,X:PHA
  370 LDA ctab+1,X:PHA:RTS
  380 .comm4 LDA (&F2),Y:INY:CMP #ASC"."
  390 BNE nextc:.comm5 INX:LDA ctab,X
  400 BPL comm5:BMI comm3:.nextc DEX
  410 .nextc2 INX:LDA ctab,X
  420 BPL nextc2:INX:INX:LDA ctab,X
  430 BNE nextc3:PLA:TAY:LDX &F4
  440 LDA #4:RTS
  450 .nextc3 PLA:PHA:TAY:JMP comm2
  460 :
  470 .ctab EQUS "BRMOVE"
  480 EQUB (move-1) DIV &100
  490 EQUB (move-1) MOD &100
  500 EQUS "BRCOPY"
  510 EQUB (copy-1) DIV &100
  520 EQUB (copy-1) MOD &100
  530 EQUB 0
  540 :
  550 .lfind STY ytemp:JSR setptr
  560 LDY #0:.lfind2 LDA (lptr),Y
  570 BMI lfind5:STA temp+1:INY
  580 LDA (lptr),Y:STA temp:LDA lin
  590 SEC:SBC temp:LDA lin+1
  600 SBC temp+1:BCC lfind3:LDA lptr
  610 STA min:LDA lptr+1:STA min+1
  620 .lfind3 SEC:LDA temp:SBC lin
  630 LDA temp+1:SBC lin+1:BCC lfind4
  640 LDA lptr:STA max:LDA lptr+1
  650 STA max+1:.lfind5 LDY #0
  660 LDA (min),Y:STA bline+1:INY
  670 LDA (min),Y:STA bline:INY
  680 LDA (min),Y:CLC:ADC min:STA min
  690 BCC lfind6:INC min+1
  700 .lfind6 LDY ytemp:RTS
  710 .lfind4 JSR nline:JMP lfind2
  720 :
  730 .nline LDY #2:LDA (lptr),Y
  740 CLC:ADC lptr:STA lptr:BCC nline2
  750 INC lptr+1:.nline2 LDY #0:RTS
  760 :
  770 .sskp DEY:.sskp2 INY:LDA (&F2),Y
  780 CMP #ASC" ":BEQ sskp2:RTS
  790 :
  800 .nget LDA #0:STA lin:STA lin+1
  810 .nget2 LDA (&F2),Y:CMP #&3A
  820 BCS nget3:CMP #&30:BCC nget3
  830 AND #&F:PHA:LDA lin+1:PHA
  840 LDA lin:PHA:ASL lin:ROL lin+1
  850 BCS toobig:ASL lin:ROL lin+1
  860 BCS toobig:PLA:ADC lin:STA lin
  870 PLA:ADC lin+1:STA lin+1
  880 BCS toobig:ASL lin:ROL lin+1
  890 BCS toobig:PLA:ADC lin:STA lin
  900 BCC nget25:INC lin+1
  910 .nget25 LDA lin+1:BMI toobig
  920 INY:BNE nget2:.nget3 JSR sskp
  930 CMP #ASC",":BNE nget4:INY
```

```
 940 JSR sskp:.nget4 RTS
 950 .toobig LDX #0
 960 .error LDA #0:STA &100:LDA err,X
 970 STA &101:INX:LDY #0
 980 .error2 LDA err,X:STA &102,Y
 990 INX:INY:CMP #0:BNE error2:JMP &100
1000 :
1010 .err EQUB 100
1020 EQUS "Line number too large"
1030 EQUB 0
1040 .bad EQUB 101
1050 EQUS "Bad resequence"
1060 EQUB 0
1070 .syn1 EQUB &DC
1080 EQUS "Syntax: *BMOVE [<start>] [<e
nd>] [<destination>]"
1090 EQUB 0
1100 .syn2 EQUB &DC
1110 EQUS "Syntax: *BCOPY [<start>] [<e
nd>] [<destination>]"
1120 EQUB 0
1130 .berr EQUB &FE
1140 EQUS "Bad command"
1150 EQUB 0
1160 .nor EQUB 102
1170 EQUS "Not enough room"
1180 EQUB 0
1190 :
1200 .move LDA #&FF:STA type
1210 LDX #syn1-err:BNE cdo
1220 :
1230 .copy LDA #0:STA type
1240 LDX #syn2-err:.cdo JSR sskp
1250 BCS cdo2:JMP error
1260 .cdo2 STY ytemp:JSR setptr
1270 LDY #0:LDA (lptr),Y:BPL notem
1280 JMP cout2:.notem JSR flen:LDA #&84
1290 JSR osbyte:STX him:STY him+1
1300 LDY ytemp:LDA (&F2),Y:CMP #ASC","
1310 BNE getst:INY:LDA #0:STA lin
1320 STA lin+1:BEQ getst2
1330 .getst JSR nget:.getst2 JSR lfind
1340 LDA max:STA start:LDA max+1
1350 STA start+1:LDA #&FF:STA lin
1360 LSR A:STA lin+1:JSR sskp
1370 BCC geten2:CMP #ASC",":BNE geten
1380 INY:BNE geten2:.geten JSR nget
1390 .geten2 JSR lfind:LDA min
1400 STA end:LDA min+1:STA end+1
1410 LDA #&FF:STA lin:LSR A:STA lin+1
1420 JSR sskp:BCC getdst:JSR nget
1430 .getdst JSR lfind:LDA min
1440 STA dest:LDA min+1:STA dest+1
1450 JSR sskp:BCC cok:LDX #berr-err
1460 JMP error
1470 .cok LDA dest:CMP start:LDA dest+1
1480 SBC start+1:BCC rok:LDA end
1490 CMP dest:LDA end+1:SBC dest+1
1500 BCC rok:LDX #bad-err:JMP error
1510 .rok SEC:LDA end:SBC start
1520 STA size:LDA end+1:SBC start+1
1530 STA size+1:SEC:LDA him:SBC pend
1540 STA temp:LDA him+1:SBC pend+1
1550 CMP size+1:BCC norr:BNE nochk
1560 LDA temp:CMP size:BCS nochk
1570 .norr LDA type:BEQ e1:JMP bits
1580 .e1 LDX #nor-err:JMP error
1590 .nochk LDA him:STA btop
1600 LDA him+1:STA btop+1:JSR shunt
1610 :
1620 .cout JSR renum:JSR flen:LDA pend
1630 STA &12:STA 0:STA 2:LDA pend+1
1640 STA &13:STA 1:STA 3
1650 .cout2 PLA:TAY:LDX &F4:LDA #0:RTS
1660 :
1670 .bits LDA #ramtop MOD &100
1680 STA btop:LDA #ramtop DIV &100
1690 STA btop+1:JSR shunt:JMP cout
1700 :
1710 .bm LDA bd:CMP bs:LDA bd+1
1720 SBC bs+1:BCS bmup:LDY #0
1730 .bm3 LDA (bs),Y:STA (bd),Y
1740 JSR dcnt:BNE bm4:RTS
1750 .bm4 INY:BNE bm3:INC bd+1:INC bs+1
1760 BNE bm3:.bmup CLC:LDA bl:ADC bs
1770 STA bs:LDA bl+1:ADC bs+1:STA bs+1
1780 CLC:LDA bl:ADC bd:STA bd:LDA bl+1
1790 ADC bd+1:STA bd+1:LDY #0
1800 .bmup2 LDA bs:BNE bmup3:DEC bs+1
1810 .bmup3 DEC bs:LDA bd:BNE bmup4
1820 DEC bd+1:.bmup4 DEC bd:LDA (bs),Y
1830 STA (bd),Y:JSR dcnt:BNE bmup2:RTS
1840 :
1850 .dcnt LDA bl:BNE dcnt2:DEC bl+1
1860 .dcnt2 DEC bl:LDA bl:ORA bl+1:RTS
1870 :
1880 .flen JSR setptr:LDY #0
1890 .efind LDA (lptr),Y:BMI efnd
1900 LDY #2:LDA (lptr),Y:CLC:LDY #0
1910 ADC lptr:STA lptr:BCC efind
1920 INC lptr+1:BCS efind
1930 .efnd LDA lptr:CLC:ADC #1
1940 STA pend:LDA lptr+1:ADC #0
1950 STA pend+1:RTS
1960 :
1970 .shunt SEC:LDA btop:SBC size
1980 STA bd:LDA btop+1:SBC size+1
1990 STA bd+1:LDA size:STA bl
2000 LDA size+1:STA bl+1:LDA start
2010 STA bs:LDA start+1:STA bs+1
2020 JSR bm:BIT type:BPL nodel
2030 LDA start:STA bd:LDA start+1
```

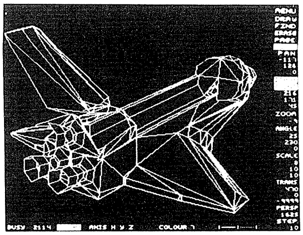
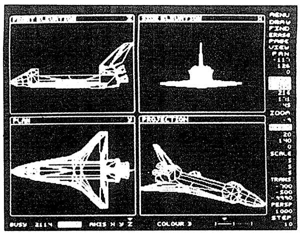
```
2040 STA bd+1:LDA end:STA bs:LDA end+1
2050 STA bs+1:SEC:LDA pend:SBC end
2060 STA bl:LDA pend+1:SBC end+1
2070 STA bl+1:JSR bm:SEC:LDA pend
2080 SBC size:STA pend:LDA pend+1
2090 SBC size+1:STA pend+1:LDA start
2100 CMP dest:LDA start+1:SBC dest+1
2110 BCS nodel:SEC:LDA dest:SBC size
2120 STA dest:LDA dest+1:SBC size+1
2130 STA dest+1:.nodel LDA dest
2140 STA bs:CLC:ADC size:STA bd
2150 LDA dest+1:STA bs+1:ADC size+1
2160 STA bd+1:SEC:LDA pend:SBC dest
2170 STA bl:LDA pend+1:SBC dest+1
2180 STA bl+1:JSR bm:SEC:LDA btop
2190 SBC size:STA bs:LDA btop+1
2200 SBC size+1:STA bs+1:LDA size
2210 STA bl:LDA size+1:STA bl+1
2220 LDA dest:STA bd:LDA dest+1
2230 STA bd+1:JSR bm:RTS
2240 :
2250 .renum JSR setptr
2260 LDA #progend MOD &100:STA min
2270 LDA #progend DIV &100:STA min+1
2280 .renum2 LDY #1:LDA (lptr),Y
2290 STA (min),Y:DEY:LDA (lptr),Y
2300 STA (min),Y:BMI renumd:LDA min
2310 CLC:ADC #2:STA min:BCC renum3
2320 INC min+1:.renum3 JSR nline
2330 JMP renum2:.renumd LDA dest
2340 STA lptr:LDA dest+1:STA lptr+1
2350 .renumd2 CLC:LDA bline:ADC #10
2360 STA bline:LDA bline+1:ADC #0
2370 STA bline+1:LDY #0:LDA (lptr),Y
2380 BMI renumd3:LDA bline+1
2390 STA (lptr),Y:INY:LDA bline
2400 STA (lptr),Y:JSR nline:JMP renumd2
2410 .renumd3 JSR setptr:.rloop LDY #0
2420 LDA (lptr),Y:BPL rloop2:RTS
2430 .rloop2 STA curl+1:INY
2440 LDA (lptr),Y:STA curl:INY
2450 .rloop3 INY:LDA (lptr),Y:CMP #13
2460 BEQ rloop4:CMP #&8D:BNE rloop3
2470 JSR change:JMP rloop3
2480 .rloop4 JSR nline:JMP rloop
2490 :
2500 .change INY:LDA (lptr),Y:ASL A
2510 ASL A:TAX:AND #&C0:INY
2520 EOR (lptr),Y:STA lin:INY:TXA
2530 ASL A:ASL A:EOR (lptr),Y
2540 STA lin+1:STY ytemp:LDA lptr
2550 PHA:LDA lptr+1:PHA:JSR setptr
2560 LDA #progend MOD &100:STA min
2570 LDA #progend DIV &100:STA min+1
2580 .change2 LDY #0:LDA (min),Y
2590 BMI nof:CMP lin+1:BNE change3
2600 INY:LDA (min),Y:CMP lin
2610 BEQ change5:.change3 LDA min:CLC
2620 ADC #2:STA min:BCC change4
2630 INC min+1:.change4 JSR nline
2640 JMP change2
2650 .change5 LDA (lptr),Y:STA lin:DEY
2660 LDA (lptr),Y:STA lin+1:PLA
2670 STA lptr+1:PLA:STA lptr
2680 LDY ytemp:LDA lin+1:AND #&3F
2690 ORA #&40:STA (lptr),Y:LDA lin
2700 AND #&3F:ORA #&40:DEY
2710 STA (lptr),Y:LDA lin+1:AND #&C0
2720 LSR A:LSR A:LSR A:LSR A:STA temp
2730 LDA lin:AND #&C0:LSR A:LSR A
2740 ORA temp:EOR #&54:DEY:STA (lptr),Y
2750 INY:INY:RTS
2760 :
2770 .nof LDX #9:.nof2 LDA fmess,X
2780 JSR oswrch:DEX:BPL nof2:LDA curl
2790 STA temp:LDA curl+1:STA temp+1
2800 CLC:PHP:LDY #4:.dp LDX #&30
2810 .dp2 SEC:LDA temp:SBC tenlo,Y
2820 PHA:LDA temp+1:SBC tenhi,Y
2830 BCC dp3:STA temp+1:PLA
2840 STA temp:INX:BCS dp2
2850 .dp3 PLA:TXA:CMP #&30:BEQ dp4
2860 PLP:SEC:BCS dp5
2870 .dp4 PLP:BCS dp5:PHP:BCC dp6
2880 .dp5:PHP:JSR oswrch
2890 .dp6 DEY:BPL dp:JSR osnewl:PLP
2900 LDY ytemp:INY:INY:INY:PLA
2910 STA lptr+1:PLA:STA lptr:RTS
2920 :
2930 .setptr LDA #1:STA lptr:LDA page
2940 STA lptr+1:RTS
2950 :
2960 .fmess EQUS " ta deliaF"
2970 :
2980 .tenhi
2990 EQUB 1 DIV 256
3000 EQUB 10 DIV 256
3010 EQUB 100 DIV 256
3020 EQUB 1000 DIV 256
3030 EQUB 10000 DIV 256
3040 .tenlo
3050 EQUB 1 MOD 256
3060 EQUB 10 MOD 256
3070 EQUB 100 MOD 256
3080 EQUB 1000 MOD 256
3090 EQUB 10000 MOD 256
3100 :
3110 .progend
3120 ]NEXT
3130 OSCLI ("SAVE RESEobj "+STR$~code+"
"+STR$~O%)
```

B

SILICON VISION

SOFTWARE FOR THE ARCHIMEDES & BBC

SolidCAD

The ultimate 3D Draughting System for Architectural design, Interior design, Engineering Design and Teaching CDT. Allows drawing in plan, front & side elevations and also directly in 3D view. Includes powerful zoom & pan options for precision draughting and surface definition for creating solid colour objects. Also includes Sweep, Extrude & Macro facilities for designing very complex objects easily. Designs created with SolidCAD are compatible with the Realtime Graphics Language for high-speed flicker-free animation. The custom Archimedes version also performs smooth shading for realism. SolidCAD(Arc) users can upgrade to the Realtime Solids Modeller (Arc) for £40.00.

£49.95 (ARC & BBC) New

REALTIME SOLIDS MODELLER

The package includes both the sophisticated design environment of SolidCAD and the high speed animation capability of a Realtime Graphics Language (RGL) module developed in pure ARM Risc code for supercharged performance. The package is ideal for Architectural design, Interior design, Engineering design & teaching CDT. The RGL module can be used to create standalone flicker-free animation of designs from your own programs. Smooth shading is also performed for realistic images. Through our in-house expertise in 3D Design and High-speed techniques, no other package can rival the design environment & animation speed of the Realtime Solids Modeller.

£89.95 (ARC) New

REALTIME GRAPHICS LANGUAGE

The Realtime Graphics Language rom provides a complete 3D Solids/Wireframe animation system with 52 star commands and 3D Editors for designing objects to animate from your own programs. Includes a 35,000 pixels / sec line generator for fast 3D drawing rates, 3D Rotate, Scale, Orbit, Perspective and Turtlegraphics. Also compatible with designs created with SolidCAD (BBC).

£49.95 (BBC)

SUPER-DUMP

The ultimate printer driver which takes advantage of the highest resolution capability of ordinary Epson compatible printers to provide 1920 × 1024 resolution. Images can also be scaled, positioned and previewed before printing. Fully compatible with SolidCAD, Realtime Graphics Language, Gate-Array design system & 3D CAD/Animation system. Your own graphics programs or other CAD packages can be made compatible with Super-Dump by the addition of a few simple commands. An example program is included in the package.

£15.95 (BBC), £24.95 (ARC) New

Presentation Manager

The package provides an interactive environment to create, edit and play-back computer controlled presentations for lectures and demonstrations. Also handles graph plotting for polynomials & user-defined functions which can be incorporated within the presentations.

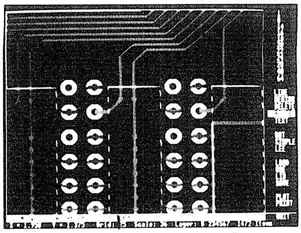
£34.95 (BBC), £49.95 (ARC) New

SILICON VISION LTD, SIGNAL HOUSE, LYON ROAD, HARROW MIDDLESEX HA1 2AG. TEL: 01-422 2274 or 01-861 2173 FAX: 01-427 5169. TELEX: 918266 SIGNAL G.

(Access/Mastercard/Eurocard accepted)

All prices include VAT and Carriage (Overseas orders add £4).

ARC-PCB



The ultimate PCB design system developed specifically for the Archimedes with a specification that cannot be matched. Includes Automatic routing, Rats-nesting, 8 layers, Surface mount capability, 0.001" resolution, 32" × 32" maximum board size, On-line Help, Fast Zoom/Pan/Redraw, Text & Silkscreen facility, Variable Line/Pad/Text/Grid sizes, Part Libraries, Block Move/Copy/Rotate/Mirror/Erase options, and up to 300,000 components. For hardcopy, the system supports a large number of plotters and ordinary Epson compatible printers at very high resolutions (1920 × 1024) of 240 dots/inch for near laser quality output. An entry-level ARC-PCB system (without auto-routing) is available for £99.95 (Auto-routing upgrade: £100). Enquire about our turnkey PCB design systems complete with Colour Archimedes and/or plotters with prices from £1220 to £3909 (Inc VAT).

£195.00 (ARC) New

RiscBASIC

Supercharge your Archimedes Basic programs by compiling them automatically into pure ARM Risc code with the RiscBASIC compiler. Features include Relocatable modules, Cross reference of all variables, functions, and procedures, Floating point and Integer support, Stand alone code generator, Optimising compiler & Full runtime error handler.

£99.95 (ARC) New

RiscFORTH

A new 32-bit implementation of the FORTH-83 standard, designed to take full advantage of the ARM architecture. Features include Multi-tasking, Optimising compiler, built-in ARM assembler with floating point mnemonics, built-in Full screen Editor, File system interface, OS calls support, Floating point & Integer maths, WIMP support, Single-step debugger, Shadow screen for documentation, Block manipulation, Dictionary & Vocabulary display, Call finding and a standalone code generator.

£99.95 (ARC) New

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.

Printer Smith-Corona dot matrix (not NLQ) heavy duty printer as new condition extra long lasting ribbon £90, Clares Artroom Master edition 80T £12, Watford Dumpout 3 as new £15. Tel. (04243) 4500.

Commstar I, manual & keystrip, unwanted gift, box not opened, £20. Your Basic or Machine Code programs blown into EPROMs from £10. Send S.A.E for details. Jeff Edwards, 48 Summer Road, Thames Ditton, Surrey, KT7 0QQ.

Master 128, 40/80T DSDD disc drive plus manual, Welcome guide, user manuals 1&2, BEEBUG C and manual, advanced sideways RAM guide, utilities discs, software plus games discs, parallel printer lead. All in excellent condition, computer in its original box. £500. Tel. 01-578 0081 ext. 2347.

Acorn 65C102 "Turbo" board £80, RH video digitiser £100, both with original software and documentation. Tel. (0222) 614401 evenings.

BBC B OS1.2 DFS, NFS, View 2.1, Speech, complete internal RF screening, boxed with manuals. for £275 ono. Tel. 021-382 7809 after 5pm.

Clares Beta-Basic and Utilities discs (with manuals) for Master 128 £10. Tel. (0902) 20768 late evenings.

WANTED: WORD PACK ROM required for Acorn Atom 12K computer. Contact Reg Walker, 23 Dover House Road, London SW15 5AA.

Disc drive, Cumana 40T, SS with mains power supply, also 1770 DFS upgrade. All mint, boxed, totally unused £75. Tel. (0294) 52250 after 6pm.

BBC B 32k issue 4. OS1.2. Watford DDFS & ROM board. £250. Watford 40/80T disc drive £60. View ROM, ViewSheet ROM £30 each inc. manuals and Star/Epson printer driver on disc. Prestel/Micronet ROM and acoustic modem £30. Also, BEEBUG back issues Vols. 2, 3 & 4 (bound) £5 each. Vols. 5 & 6 (unbound) £4 each. Individual issues Vols. 1 & 7 40p each. Also "The Book of Listings" and "30 Hour Basic" £2 each, £3.50 for the two. Tel. (089 582) 3613 evenings/weekends.

BBC B, Solidisk 32K SWR, double-sided 100K disc drive 6502 second processor and plenty of software and magazines. £350. Tel. 01-340 9442.

Printwise 80T (series II), hardly used £17.50. Sprites 40T £7.50. Tel. (0302) 744005.

Shinwa CP80 printer, in excellent order £80. Updating to 24 pin. Carriage extra. Tel. (0428) 713326 evenings.

Seikosha GP100A printer, manual, cable, screen dump and full instructions. £35. Watford 32k RAM-ROM board. Fitting available, Tyne & Wear area. £16. Tel. (091) 4168998.

BEEBUG cassettes Vol.1 No.10 - Vol.5 No.7 (38 total) £20. BEEBUG discs Vol.5 No.8 - Vol.6 No.10 (13 total) £25. Magscan bibliography plus 4/5 update £9. Dumpmaster ROM £12, ACP ROM Manager £8, Printwise (disc) £10, Masterfile disc (ADFS) £8, BEEBUG starter pack (cassette) £5, View/ViewSheet user guides £4 each, Colossus Chess 4.0 (2 discs/manual) £6, Revs + 4 tracks £6. Akhter 40/80T disc drive 400k(640k) powered from computer £55, Epson LX80 tractor feed plus LX80 manual £15, Voltmace Delta 14b and 14b/1 interface £13. Tel. (0236) 723615.

BBC issue 7 plus 1770 plus ADFS £295, 400k DS disc drive £90, Acorn twin joysticks £7, Acorn cassette recorder £16, Zenith green screen monitor £40, Aries B20/B12 complete £55, Aries B32, B12 complete for £85, Tel. (0403 81) 4976.

Taxan XP915 wide cartridge dot matrix NLQ printer. Epson plus IBM emulation hardly used. £325 ono. Tel. Gloucester 730705.

Archimedes 310 colour system with BEEBUG disc interface, Acorn backplane and Computer Concepts ROM/RAM board, as new, under warranty £900. Tel. (0324) 558692.

BBC B, Z80 second processor, dual DS disc drive, auto-dial modem. colour monitor with View, Wordwise etc ROMs. £500. Tel. 01-534 2119.

Continued on page 37

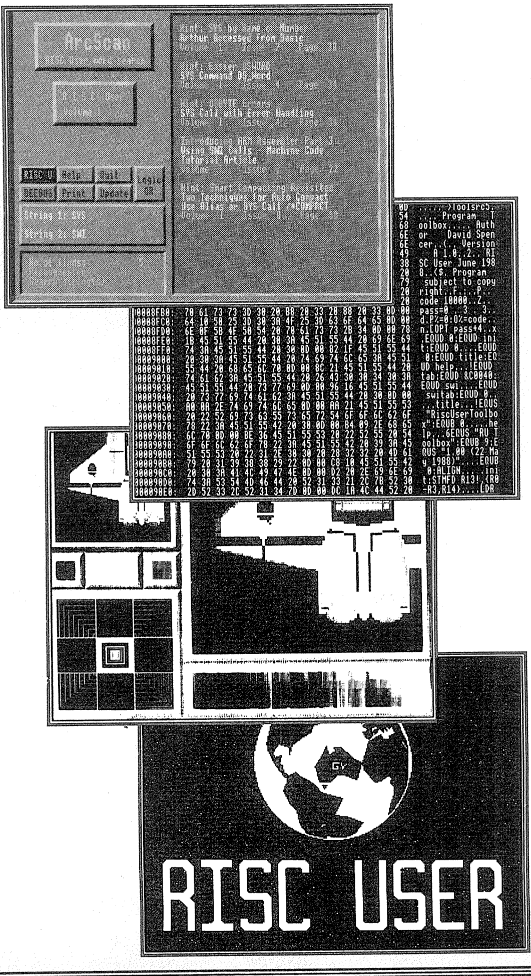
RISC USER

The Archimedes Magazine and Support Group

IS ONE YEAR OLD
GREAT BIRTHDAY OFFER
WORTH OVER £50
The volume One Special Disc
Yours for only £4.95
when you join RISC User.
SUBSCRIBE NOW AND BENEFIT FROM OUR BIRTHDAY OFFER SPECIAL RATES FOR BEEBUG MEMBERS.

Offer terminates 31 Dec 1988. Join RISC User now!

SPECIAL DISC CONTAINS ALL THIS FOR JUST £4.95



```
Arthur Accessed from Basic
SYS Command OS_Word
SYS Call with Error Handling
Using SWI Calls - Machine Code
Tutorial Article
Two Techniques for Auto Compact
Use Alias or SYS Call /*COMPACT
```

```
                                                 0D  .....)Toolsrc5.
                                                 54  .... Program   T
                                                 68  oolbox..... Auth
                                                 6E  or    David Spen
                                                 6E  cer..(.. Version
                                                 49     A 1.0..2.. RI
                                                 38  SC User June 198
                                                 20  8..($. Program
                                                 79   subject to copy
                                                 20  right..F.:..P..
                                                 20  code 10000..Z..
0008FB0:  70 61 73 73 3D 30 20 B8 20 33 20 88 20 33 0D 00  pass=0 . 3 . 3..
0008FC0:  64 10 50 25 3D 30 3A 4F 25 3D 63 6F 64 65 0D 00  d.P%=0:O%=code..
0008FD0:  6E 0F 5B 4F 50 54 20 70 61 73 73 2B 34 0D 00 78  n.[OPT pass+4..x
0008FE0:  1B 45 51 55 44 20 30 3A 45 51 55 44 20 69 6E 69  .EQUD 0:EQUD ini
0008FF0:  74 3A 45 51 55 44 20 30 0D 00 82 1F 45 51 55 44  t:EQUD 0....EQUD
0009000:  20 30 3A 45 51 55 44 20 74 69 74 6C 65 3A 45 51   0:EQUD title:EQ
0009010:  55 44 20 68 65 6C 70 0D 00 8C 21 45 51 55 44 20  UD help...!EQUD 
0009020:  74 61 62 3A 45 51 55 44 20 26 43 30 30 34 30 3A  tab:EQUD &C0040:
0009030:  45 51 55 44 20 73 77 69 0D 00 96 16 45 51 55 44  EQUD swi....EQUD
0009040:  20 73 77 69 74 61 62 3A 45 51 55 44 20 30 0D 00   switab:EQUD 0..
0009050:  A0 0A 2E 74 69 74 6C 65 0D 00 AA 21 45 51 55 53  ...title...!EQUS
0009060:  20 22 52 69 73 63 55 73 65 72 54 6F 6F 6C 62 6F   "RiscUserToolbo
0009070:  78 22 3A 45 51 55 42 20 30 0D 00 B4 09 2E 68 65  x":EQUB 0.....he
0009080:  6C 70 0D 00 BE 36 45 51 55 53 20 22 52 55 20 54  lp...6EQUS "RU T
0009090:  6F 6F 6C 62 6F 78 22 3A 45 51 55 42 20 39 3A 45  oolbox":EQUB 9:E
00090A0:  51 55 53 20 22 31 2E 30 30 20 28 32 32 20 4D 61  QUS "1.00 (22 Ma
00090B0:  79 20 31 39 38 38 29 22 0D 00 C8 10 45 51 55 42  y 1988)"....EQUB
00090C0:  20 30 3A 41 4C 49 47 4E 0D 00 D2 20 2E 69 6E 69   0:ALIGN... .ini
00090D0:  74 3A 53 54 4D 46 44 20 52 31 33 21 2C 7B 52 30  t:STMFD R13!,{R0
00090E0:  2D 52 33 2C 52 31 34 7D 0D 00 DC 1A 4C 44 52 20  -R3,R14}....LDR 
```

1. ARCSCAN
   A fast on-screen bibliography with powerful search facilities for all the RISC User and BEEBUG magazines. Normal price £12.

2. PIXEL EDITOR
   This powerful drawing tool is a full screen full-feature pixel editor for creating and editing screens and sprites.

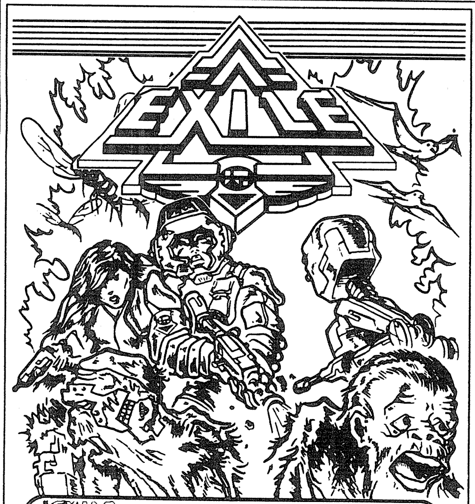
3. TOOLBOX
   This incredibly useful utility features a memory editor, memory search and replace, disc editor and disassembler. TOOLBOX contains many of the features found in packages costing over £35.

4. WORLD IN MOTION
   A stunning animation with an oddly reminiscent feel to it.

5. DISC MENU MODULE
   Use the mouse to control your disc files with this extremely useful relocatable module.

6. PRINTER BUFFER
   This printer buffer frees your computer during long printouts and is configurable from a few bytes to 4 Mbytes. Similar to packages currently selling at £19.

Altogether the items on the disc would be worth over £50 if bought separately. See page 70 for reduced subscription rates for BEEBUG members.



"EXILE is an even more enjoyable game to play than ELITE or ZARCH"

... David Braben (co - author of Elite, author of Zarch).

BBC Micro Cassette	£12. 95	Acorn Electron Cassette	£12.95
BBC Micro 5¼" Disc	£14.95	BBC Master Compact 3½" Disc.	£19.95

The Ultimate Repton - four puzzling new games and a complete games designer.

BBC Micro Cassette	£12.95	Acorn Electron Cassette	£12.95
BBC Micro 5¼" Disc	£14.95	BBC Master Compact 3½" Disc	£19.95

BECOME THE WORLD CHAMPION — BY FAIR MEANS OR FOUL

Fair Moves: Head Punch, Body Blow, Upper Cut, Duck Punch
Foul Moves: Head Butt, Knee, Groin Punch, Kick

BBC Micro Cassette	£9.95	Acorn Electron Cassette	£9.95
BBC Micro 5¼" Disc	£11.95	BBC Master Compact 3½" Disc.	£14.95

FOUR-GAME COMPILATIONS FOR THE BBC MICRO AND ELECTRON

VARIETY WITH GREAT VALUE FOR MONEY

Play It Again Sam 3

COMMANDO (Elite), PALACE OF MAGIC (Superior Software), KILLER GORILLA (Micro Power), KILLER GORILLA 2 (Superior Software).

Play It Again Sam 4

FRAK! (Aardvark), SPELLBINDER (Superior Software), COSMIC CAMOUFLAGE (New Release), GRAND PRIX CONSTRUCTION SET (On the BBC Micro version - Superior Software), GUARDIAN (On the Electron version - Alligata).

Play It Again Sam 5

IMOGEN (Micro Power - Now also on the Electron), ELIXIR (Superior Software), BUG BLASTER (Alligata), FORTRESS (On the BBC Micro version - Pace/Amcom), MOONRAIDER (On the Electron version - Micro Power).

BBC Micro Cassette	£9.95	Acorn Electron Cassette	£9.95
BBC Micro 5¼" Disc	£11.95	BBC Master Compact 3½" Disc.	£14.95

SUPERIOR SOFTWARE Limited   ACORNSOFT

Access   VISA

Dept. BB1, Regent House, Skinner Lane, Leeds LS7 1AX. Telephone: (0532) 459453

Please make all cheques payable to "Superior Software Ltd".

Personal Ads (continued from page 34)

WANTED: Graphics software 'Tesselator' (including manual), as published by Addison-Wesley. Please write to IB Heide, 24 Bueager, DK-2950 Vedbak, DENMARK.

WANTED: The two software cassettes related to Angell & Jones book 'Advanced graphics with the BBC model B microcomputer'. Please write to IB Heide, 24 Bueager, DK-2950 Vedbak, DENMARK.

BBC Micro Issue 7, excellent condition, still boxed with Watford DDFSand View, User Manual and Advanced User Manual £250 o.n.o. Tel. (092385) 8202.

Watford Le Modem all complete with manuals and ROM £50. Acorn 6502 3MHz second processor £50. Tel. (0908) 614573.

Master 128 as new, Double sided 40/80T switchable, 5.25" Disc Drive, InterWord, Music 500 synthesizer with amp and speakers £500 o.n.o. Tel. (0279) 22466.

AMX Mouse with Super Art ROM, utility disc and manual £25. InterWord (Computer Concepts) ROM with manual, reference card and key strip £30. Tel. (0462) 50868.

Acorn Pascal for Master £45. Tel. (04685) 477.

BBC issue 7 OS 1.2 Acorn 1770 DFS. Cumana 40/80T double sided disc drive still under guarantee. Excellent condition. £250. Consett. Tel. (0207) 508940.

Archimedes A310 with colour monitor, Printer cable, BEEBUG serial link, 1st Word Plus, Zarch, Books, RISC USER magazine and discs, Diskettes and storage case. Offers over £800. Tel. (0438) 358519 eves.

BBC Spare Parts PSU with leads £30. Complete Acorn DFS 1.20 (8271 controller) £25, Basic II £7, x2 OS 1.20 £5 each, x2 6502 CPU £5 each, x2 Video ULA £5 each, x2 SAA 5050 Teletext £2 each, BBC Keys 50p each. All parts from issue 7 BBCs. Tel. (0454) 613977 after 4pm.

Surplus software 5.25" discs, Masterfile 2 £11. Quickcalc £9. Disc Master £10. Paintbox £7. Watford Disc Executor (tape to disc utility) £5. Games cassettes: Arcadians, Planetoid, Snapper, Chess, Draughts and Reversi, Billiards, Magic-Eel, Starfire, all for £10. 737 Flight Simulator £4. The lot £50. Tel. (0582) 460065.

Archimedes 310 colour with 2nd disc drive, Podule Backplane and CC ROM Podule £800. Archimedes software, InterWord, InterSheet, System Delta Plus, Artisan, Graphic Writer, Archimedes Toolkit Module, all at approx half current price. Also many items of BBC/Master software. Tel. (0923) 221425.

Continued on page 38

ASTAAD

This keystrip is designed to be used with new version of ASTAAD on pages 24 to 28 of this issue. Either cut out the keystrip or photo-copy this page and place under the plastic strip on your micro when using this program.

Key f0	Key f1	Key f2	Key f3	Key f4	Key f5	Key f6	Key f7	Key f8	Key f9
PRINT SCREEN	VECTOR RECT	SOLID OR DOTTED	HOME CURSOR	MIRROR IMAGE	MARGINS	SCALE	LINK STEAMS	SAVE SCREEN	LOAD SCREEN
SOFT ASTAAD	COPY OR MOVE	LINE OR ARROW	CIRCLE OR ARC	POLYGON ELLIPSE	INFILL OUTLINE	COLOUR REVERSE	LINE FIX OR TRANS	2 OR 16 STEPS	ORIGIN REL/ABS
ASTAAD	AREA MOVE	LINE	CIRCLE	POLYGON	REPEAT F2, F3, F4	MOVE	DRAW	RUBOUT	DELETE AREA

Personal Ads (continued from page 37)

BBC B issue 7 OS1.2. Opus DDOS & 40/80 DSDD disc drive replay. Watford ROM/RAM card with 16k battery backed RAM. Fleet Street Editor, Mini Office II and many books & games. Acorn Users Jan'85 - Dec'87. All boxed with leads and manuals. Bargain at 475. Tel. (05827) 61917.

WANTED. Replacement Torch second processor board (ZEP100), software & MCP ROM not essential - must be cheap. Also required STL DFDC at reasonable price. Tel. (0992) 711555 or PresTel. MBX 01111029, TGold 82:T0X093.

Master 128 £250, B+ with DDFS £180, Kaga Taxan monochrome monitor £50, Metal case Microvitec colour monitor (med res) £150, Sanyo green monitor £55. All prices o.n.o. Tel. (0236) 20199 day, (0786) 833541 eves.

Microwriter £95, Electron, Data Recorder and software £45. R.Newmark Tel. 091-536 2066 (not weekend).

Watford Sideways ZIF, Issue7modelB, Watford ROM card, Opus 8502 Disc Drive, DDFS Disc Interface, MP165 Dot Matrix Printer, High resolution colour monitor, 50 blank discs. £200 worth of software, £700 o.n.o. Tel. (0386) 750269 eves.

BBC series 7, Solidisk DDFS, 80T Morley Teletext PSU, Mini Office II ROM, Replay ROM, APL ROM board, all with manuals, some games, £375 o.n.o. Tel. (0271) 816658.

Compact with 5.25" cable, Printer cable, Dumpout, mags, games, etc. £450. Electron, ROM Box Plus, T2P3, Plus 3 Database, View, ViewSheet, games, discs, tapes, mags, books etc, £250. Tel. (092576) 4832.

Printer Epson RX-80F/T+ £100. Watford NLQ ROM £10. Watford TRANSFEROM £5. All with manuals. Tel. (0252) 523258 eves.

Aries B32 card fully documented £40 plus postage. Tel. (0932) 226076.

Beebugsoft Discmaster £14, View printer driver generator disc £5,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. (0279) 832548 eves.

Spellmaster £27 or swap for InterBase. Tel. (0536) 67041 6.30 to 8.30.

Original software for sale: Computer Concepts' Printmaster ROM £12, Computer Concepts' Disc Doctor ROM £9, BEEBUG Toolkit ROM £10, all in original packing with instructions. Tel. (0786) 61501.

BBC B Acorn DFS/8271. AMX Mouse/EPROM, Desk Utils, Art & Paint-Pot discs, User manuals. Torch Z80 processor, complete package. Perfect Filer Calc, Writer & Speller. EPROMs, discs and manuals as new. RAM/ROM Board with disc and manual. Miracle Modem WS2000 and manual. Thorn tape cassette/leads. Wide selection of BBC entry level books. Sensible offers. Tel. 01-876 5347 eves.

Dual disc drive, DS 40T, power from BBC (Qume from Microware), £125 inc. documentation, utility disc and cover. Tel. 01-444 0521.

Cumana 40T s/s disc drive own PSU and leads £30 o.n.o., Watford disc interface £20, Watford File+ Database, ROM Spell, Graphics ROM, Printmaster, Beebcalc, Sleuth, Toolkit, £10 ea. BEEBUG Spellcheck, Masterfile, Investigator, Betabase, £6 each. Disc executor £4. Disc games Emthar 7, Elite £4 each; Acorn drawing software, Graphs and Charts, Quest and loads more £2 each. All originals. Tel. (0727) 30264.

AMX mouse with Stop Press and Extra Extra for use with Master 128 £55. Tel. (0474) 363503.

Akhter 40/80T double-sided disc drive with PSU still in guarantee £80 o.n.o. Tel. (04867) 5107 eves.

Plus 1 Electron expansion unit, boxed with manual, View, a couple of ROM games. All vgc £30. Tel. (0344) 51308.

Unwanted gifts: 6 discs all Master 128 compatible, Empire Strikes Back, Uridium, Impact, Omega Orb, Trivial Pursuit, Despatch Rider plus Bulls Eye cassette. A & B Computing Master Graphics packs 1 & 2 ADFS plus instructions. A & B Computing mags from Jan 87 to present binders included. £55 or nearest sensible offer. Tel. (0705) 527957 after 6pm.

BBC B, DDFS, ADFS, Toolkit, Wordwise Plus, Comstar modem with autodial, Seikosha printer, Hitachi monitor/TV, software, books and cassettes £300 o.n.o. Tel. (0753) 682449.

Master Turbo, Viglen Keyboard/Case, Twin 40/80DS/DD drives, ViewStore, Pagemaker, Mouse, Cartridges, NLQ ROM, software, manuals £650. Tel. (0895) 449726.

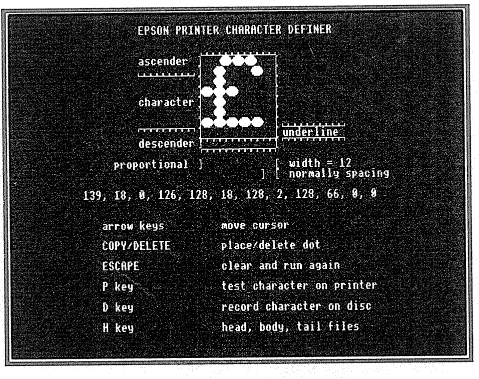
ROM Board,Watford solderlessROM/RAM Board, including 16K RAM, battery back-up and Read/Write protect switches, £22. Tel. (06845) 64106.

Continued on page 40

The Best of BEEBUG

The very best of all the programs published in BEEBUG are now available conveniently grouped together by subject on disc. The first two compilations are GENERAL UTILITIES and APPLICATIONS. In each case the disc is complete and self contained. All the programs may be loaded ready for use through a customised menu system, and all the information you need to know is included on the disc too for displaying on your screen or outputting to your printer. These discs offer exceptional value for money, and are available only to BEEBUG and RISC User members.

Each disc (5" only) costs just £5.75, plus post and packing 60p (90p for two). Simply complete the order form below, or if you wish to pay by Access, Visa or Connect just phone in your order.



```
EPSON PRINTER CHARACTER DEFINER

ascender

character

                              underline
descender

proportional ]           [ width = 12
                         ] normally spacing

139, 18, 0, 126, 128, 18, 128, 2, 128, 66, 0, 0

arrow keys        move cursor
COPY/DELETE       place/delete dot
ESCAPE            clear and run again
P key             test character on printer
D key             record character on disc
H key             head, body, tail files
```

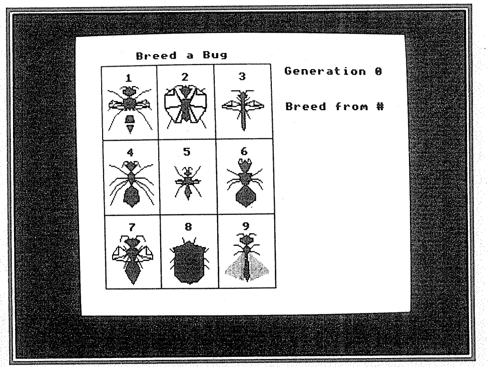
General Utilities

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

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

Applications

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:

1.General Utilities Disc Code 1605A ☐ **2.Applications Disc** Code 1604A ☐

Name ______________________ Total £________

Address ______________________ Postage £________

______________________ Grand Total £________

Memership No ______________________

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.

Personal Ads (continued from page 38)

Torch Graduate with MS DOS, 256K twin drive and Psion Xchange software with manuals £250; Taxan KX-12 an excellent mono monitor £35, BBC B Issue 4 OS 1.2 in full working order £125. Tel. (0737) 221352.

BBC Master 128 £300. Philips CM 8524 colour monitor £150. Opus Disc Drive 40T £75. Dual Joysticks £12.50. Peartree MR6000 cartridge £8.00. ViewStore ROM £30. Spellmaster ROM £40. BEEBUG Dumpmaster ROM £15. PMS GENIE, Module, Battery backup £55, BEEBUG'S PrintWise£10, Instant Recall Database on disc £5, BBC Programming course with book £5. Various games & Utility discs £10. Various BEEBUG AU discs £15. Step by Step programming books 1&2 £5.50. View, Sheet & Store User guides £20. Reference manuals 1&2 £10. Mastering View, ViewSheet & ViewStore £6. Large number of BEEBUG & AU magazines. The whole lot for £750 ono. Write to: Mr Northam, 45 Citadel Road, The Hoe, Plymouth PL1 3AU.

BEEBUG 'C' includes 2 ROMS, Stand Alone Generator and K&R book, complete with manuals £50 ono. Tel. (0384) 455066 eves.

2 Acorn Electrons in good working order £20 each. Tel. (0245) 468707.

512 Co-Processor plus Watford Co-Pro adaptor £80. Tel. (0923) 779767.

Brother HR5 Printer, with one roll of thermal paper, £35. View WP ROM & manuals £18. Vine Micros Replay for Solidisk 1770 DFS £12. Tel. 01-777 3811 eves.

WANTED: Gemini DataBase cassette for BBC micro A or B 32K. Tel. (0245) 352714 after 6pm.

Canon twin 40/80T disc drive with PSU £160. Philips 14" green screen monitor £40. Tel. (0763) 82518.

Sleuth (Basic debugger) £15, Oxford Pascal £20, Gemini DataGem (database) £30, Acorn BCPL £30, BCPL Stand Alone Generator £20, Acorn BASIC Editor £15, Hershey Fonts (80T) £8. All in original packing with complete manuals. Acorn User (Sept 82 to Jan 88 complete), The Micro User (Mar 83 to Jan 88 complete), A&B Computing (May 83 to Dec 87, only Sep 87 missing) all in excellent condition, offers invited. Tel. (051) 336 3765 eves.

Panasonic printer KX-P1081 (new) £100, Epson MX100 £40, CJE Multifont NLQ (20 fonts) £15, PMS Multifont NTQ £15, Clares Fontwise Plus £15. Other utilities and Superior games on disc (£5 each). Send SAE to A Provan, 5 Garvel Road, Milngavie, Glasgow, G62 7JD.

Triumph Addler 7020 20CPS daisy wheel printer, cable and spare ribbons £125, NEC 8023 100 CPS matrix printer £45. Tel. (0256) 471916.

BEEBUG Telesoftware Passwords

For the benefit of newer members we list below the passwords needed to access BEEBUG Telesoftware from Micronet.

Program	Password
Function Keystrip	*SAME*
Address Book	*DISTINCT*
Telephone Book	*PATTERN*
ROM Controller	*(No password needed)*
Compare	*STRING*
Disk Manager	*CATEGORY*
World by Night & Day	*MARKET*
Ghost Host	*RELEVANT*

ADVERTISING IN BEEBUG

For advertising details, please contact
Sarah Shrive
on:

(0727) 40303
or write to:

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

Triumphs and Tribulations of a Tyro

D.A.Higgs explains, for the benefit of other newcomers to this system, how he has learnt to organise his Master.

Whether being in my seventies I can be classed as a tyro is, maybe, a moot point but it is irrelevant to the present discourse. When I retired from my company-provided office, and no longer had an indefatigable amanuensis on the other side of the door, yet still had a diversity of matters requiring my attention, I decided there was a need to do something about it. On the basis that, as it appears possible to drive a car without being a Chartered Engineer, it should be feasible for someone who is computer illiterate to drive a computer, I ventured upon the purchase of a BBC B, a RX-80F/T printer and a pair of Cumana double sided 80 track 5.25" disc drives, together with a Wordwise Plus ROM.

Disc Structure				Interpretation
$				Root Directory
	FKEYS			Function Key Definitions
	PD1			Printer Driver for View
	STARTER			BEEBUG Starter Program
	USERDIC			Spellmaster user dictionary
	VPD			View source file for printer driver
	WP			Word Processing Directory
		ADS		Addresses Directory
			ADD1	Sample address 1
			ADD2	Sample address 2
		FORM		Letter formats
			A4	A4 size
			A5	A5 size
	SS			Spreadsheet Directory
		PBV		Sample spreadsheet 1
		TDH		Sample spreadsheet 2
	GRPH			Graphs Directory
		(no entries)		

Figure 1. Suggested Disc Structure

Recently, having suffered a technical failure (of the power pack) of my model B, I elected to upgrade to a Master 128. This provided new tricks in the shape of View, View Sheet and the ADFS. I also installed Computer Concepts' Spellmaster and Watford Electronic's NLQ ROMs.

I was further motivated by the *STARTER* program included in BEEBUG Vol.4 No.8 Page 31 to see what I could do in these contexts to make the job of retirement easier.

Whether the outcome of my trials and tribulations, in producing such as I have managed to achieve, has any interest or benefit for others I leave you, the reader, to judge. I only know it is of assistance to me.

I have gradually developed an 'initialisation' disc (ADFS format) containing a !BOOT file and various other files which I use to set up my machine on pressing Shift-Break (or the Master could be configured to boot this disc automatically on power-up). The disc is structured as shown in figure 1. The meaning of these files and directories will become clear in due course.

First of all I created a !BOOT file so that it calls up the Spellmaster User Dictionary *USERDIC* (this takes a short time to load but at least you don't forget it), followed by *FKEYS*, and then an adaptation of the *STARTER* program. The function key definitions (see listing 1) are mostly used for word processing.

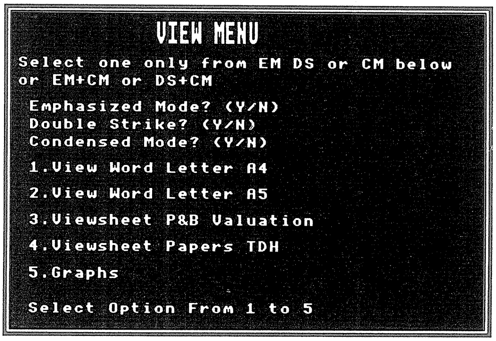
Listing 1

```
   10 REM Program FKEYS
   20 REM If words are to be added to US
ERDIC, use f6 before keys f8 & f9
   30 REM If envelope is to be addressed
 from ADS directory use f7 before keys f
8 or f9
   40 REM For commands press Shift+Ctrl+
Function Key in Command Mode
   50 REM For inclusion in letters press
 Shift+Ctrl+Function Key in Edit Mode
   60 *FX228,1
   70 *KEY1 Membership No.
   80 *KEY2 Yours faithfully,
   90 *KEY3 Yours sincerely,
  100 *KEY4 Kind regards,
  110 *KEY5 D.A.Higgs.
  120 *KEY6 *DIR $|M*DSAVE|M
  130 *KEY7 *DIR $.WP.ADS|M
  140 REM Set marker where file is to be
 inserted & call with READ <filename> 1
```

```
150 *KEY8 *NLQ80|M
160 *KEY9 *NLQTYPE|M
170 END
```

Thereafter, using Ctrl-Shift-f6 saves "USERDIC" in the root directory $, with any additions that have been made on checking the draft document. This should always be done before selecting NLQ mode on the printer, or the computer may hang. If you do not want to employ NLQ, you can still save the updated dictionary without disturbing, say, the Emphasised Mode already selected from the Menu (see figure 2).

This is because to activate the NLQ mode it is necessary to enter, in command mode, *NLQTYPE which can now be achieved with Ctrl-Shift-f9. In the interest of speed, any printing for editing or file copy purposes should be carried out in the default draft mode before pressing f9 as above.



```
VIEW MENU
Select one only from EM DS or CM below
or EM+CM or DS+CM
 Emphasized Mode? (Y/N)
 Double Strike? (Y/N)
 Condensed Mode? (Y/N)
 1.View Word Letter A4
 2.View Word Letter A5
 3.Viewsheet P&B Valuation
 4.Viewsheet Papers TDH
 5.Graphs
 Select Option From 1 to 5
```

Figure 2. Screen Menu

The *STARTER* program is shown as Listing 2. It displays the Menu (figure 2) from which a choice has first to be made as to the style of printing (Emphasised, Double Strike or Condensed or indeed certain possible combinations). Then the choice is between letter heads of A4 or A5 format. I find that most letters can be accommodated on a single sheet (I use single sheets for quality reasons), and so these are set up as in figure 3.

Listing 2

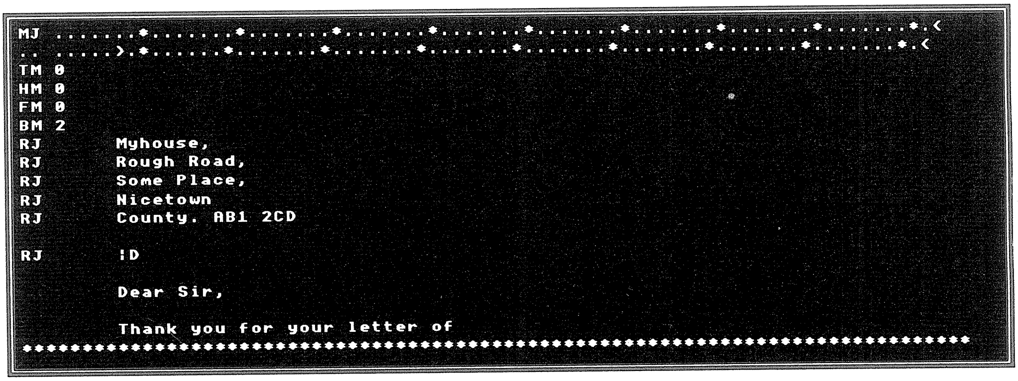
```
   10 REM Program STARTER
   20 REM Version D2.0
   40 REM BEEBUG Jan/Feb 86 Modified
   50 ON ERROR REPORT:PRINT" at line ";E
RL:I=INKEY(300):MODE 0:END
   60 :
  100 MODE7
  110 FOR I=1 TO 2:PRINTTAB(0,2+I)CHR$13
2;CHR$157;TAB(10)CHR$141;CHR$134;"VIEW M
ENU":NEXT I
  115 PRINT "Select one only from EM DS
or CM below  or EM+CM or DS+CM";
  120  PRINT''CHR$134;"Emphasized Mode?
(Y/N)";
  130 P$=GET$
  140 IF P$="Y" VDU 2,1,27,1,69,3
  142 PRINT'CHR$134;"Double Strike? (Y/N
)";
  143 P$=GET$
  144 IF P$="Y" VDU 2,1,27,1,71,3
  145 PRINT'CHR$134;"Condensed Mode? (Y/
N)";
  146 P$=GET$
  147 IF P$="Y" VDU 2,1,15,3
  150 PRINT''CHR$134;"1.View Word Letter
 A4"
  160 PRINT'CHR$134;"2.View Word Letter
A5"
  170 PRINT'CHR$134;"3.Viewsheet P&B Val
uation"
  180 PRINT'CHR$134;"4.Viewsheet Papers
TDH"
  190 PRINT'CHR$134;"5.Graphs"
  200 PRINT''CHR$130;"Select Option From
 1 to 5 ";
  210 A=GET-48
  220 IF A=1 THEN PROCone
  230 IF A=2 THEN PROCtwo
  240 IF A=3 THEN PROCthree
  250 IF A=4 THEN PROCfour
  260 IF A=5 THEN PROCfive
  500 *FX138,0,128
  510 END
  520 :
 1000 DEFPROCone
 1020 *KEY0 *WORD|M MODE 131|MNEW|MLOAD
$.WP.FORM.A4|M
 1030 ENDPROC
 1040 DEFPROCtwo
 1060 *KEY0 *WORD|M MODE 131|MNEW|MLOAD
$.WP.FORM.A5|M
 1070 ENDPROC
 1080 DEFPROCthree
 1090 *KEY0 *SHEET|M MO.131|MLOAD $.SS.P
BV|M
 1100 ENDPROC
 1110 DEFPROCfour
 1120 *KEY0 *SHEET|M MO.131|MLOAD $.SS.T
DH|M
 1130 ENDPROC
 1140 DEFPROCfive
```

```
1150 REM DEFINE GRAPH ACTION WHEN KNOWN
1170 ENDPROC
```

If more than one sheet becomes necessary, then the Footer Margin stored command needs to be changed to FM 1, and the Footer enabled with the additional stored commands FO ON. The Footer itself is then defined as, for example, DF / /page |p/ /.

the ADS directory using function key f7 before selecting NLQ mode, if needed.

If neither of the letter heads is wanted, then by selecting option 3 or 4, either the Spreadsheet PBV (Portfolio and Bond Value) or TDH (T and D Helpers) is loaded. Of course, you can easily modify the *STARTER* program to select any View or ViewSheet file you wish.



```
MJ  .......*.........*.........*.........*.........*.........*.........*.........*........*.<
..  ......>.*.........*.........*.........*.........*.........*.........*.........*........*.<
TM 0
HM 0
FM 0
BM 2
RJ       Myhouse,
RJ       Rough Road,
RJ       Some Place,
RJ       Nicetown
RJ       County. AB1 2CD

RJ       |D

         Dear Sir,

         Thank you for your letter of
**************************************************************************************************
```

Figure 3. A4 Letter Format

The word processing activities are all kept within the directory WP, which itself is in the root directory $. There are two sub-directories in WP called Form and ADS. The former contains the files A4 and A5, and the latter the files ADD1 and ADD2. Any of the two addresses ADD1 and ADD2, as an illustration, can be entered on whichever letter heading has been loaded in View by pressing Ctrl-Shift-f7 to get to the ADS directory, marking the place the address is to appear in the letter with, say, marker 1, and then in command mode typing:

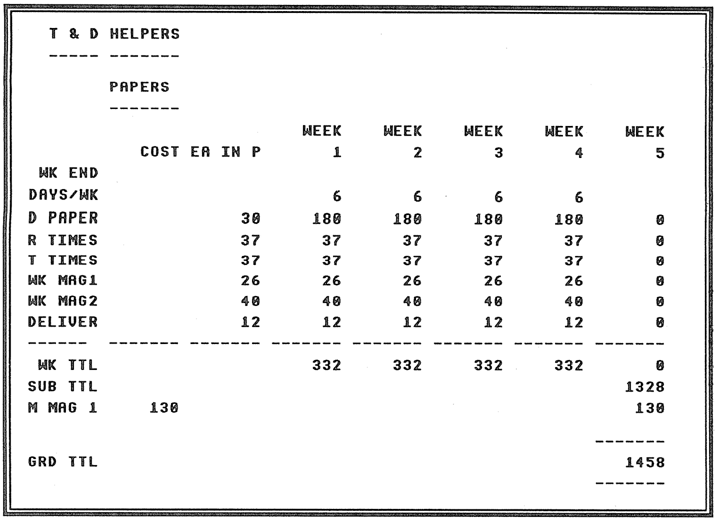
```
    READ ADD1 1.
```

To print the address of the addressee on an envelope of size 6"x3.5", put it in the printer as far to the left as it will go, and then line up the top of the envelope with the top of the ribbon mask, open the scale and then, in the command mode, check you are in the ADS directory and type in P ADD1. When the top of the envelope is above the scale, return it to its position against the envelope before printing commences. Again, I find that you need to enter

The purpose of TDH, which is shown in figure 4, is to check the newsagent's (T and D Helpers) monthly bill. Because I am away from time to time this varies accordingly.

If the price of any publication changes then entering the new price in the COST EA column will alter the price in the WEEK columns. Changing the number of days per week in row DAYS/WK for any of the four weeks will will cause the cost of the D(aily) PAPER and DELIVER(Y CHARGE) for that week to adjust pro rata. When a five week month comes round, putting the appropriate number of days under WEEK 5 in row DAYS/WK will fill in the other cells in this column and include the extra cost in the total. Using this spreadsheet, I find I get the total right more often than the newsagent!

The GRAPH option offered in the menu is to provide for something else - perhaps a graph - to be added later. It will, of course, necessitate altering lines 190 and 1150 of the *STARTER* program.



```
  T & D HELPERS
  ----- -------

        PAPERS
        -------
                          WEEK    WEEK    WEEK    WEEK    WEEK
           COST EA IN P     1       2       3       4       5
 WK END
DAYS/WK                     6       6       6       6
D PAPER            30     180     180     180     180       0
R TIMES            37      37      37      37      37       0
T TIMES            37      37      37      37      37       0
WK MAG1            26      26      26      26      26       0
WK MAG2            40      40      40      40      40       0
DELIVER            12      12      12      12      12       0
------ ------- ------- ------- ------- ------- ------- -------
 WK TTL                   332     332     332     332       0
SUB TTL                                              1328
M MAG 1     130                                       130

                                                  -------
GRD TTL                                              1458
                                                  -------
```

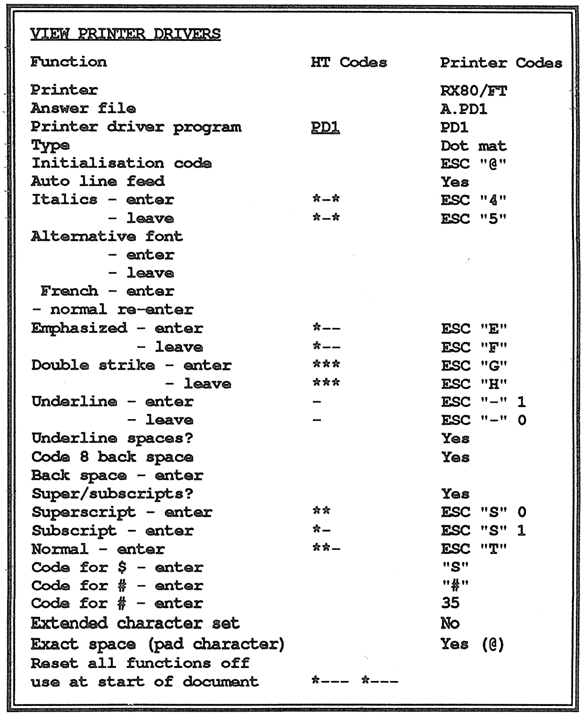
Figure 4. TDH Spreadsheet

In the root directory $ there are two more files not yet mentioned. PD1 is a printer driver generated by Acornsoft's Printer Driver Generator disc. The other file, VPD, was written in View, and as can be seen in figure 5, it lists the codes used in PD1. I also generated another driver PD2 (not illustrated here), to make use of the French font available with my printer. Although it called up the font correctly, I could not make the printer backspace to place an accent over a letter as is possible in Wordwise using the embedded command OC8. Does anyone else have any bright ideas?

Although I had understood that the default values for the Highlights (in View) will produce *underline* and *bold*, it was not until after many trials and tribulations, because as a tyro I naturally assumed it was my fault that it would not work, that I discovered that without a printer driver the Highlights will not work at all on the Master - another reason for embarking on my efforts with STARTER and PD1 etc.

I recognise that all this is elementary to any experienced user, and probably not the most elegant achievement, but it at least has the merit of working! I would have been delighted to have had something akin to this to copy (blindly if you like) to get me going, and to have worked out the why and wherefore at leisure afterwards.

I trust that others new to the Master (and Compact) will be able to learn from my own experiences in this field. packaging up your micro with a customised boot file system certainly makes life a lot, lot easier for regular use of View and ViewSheet.



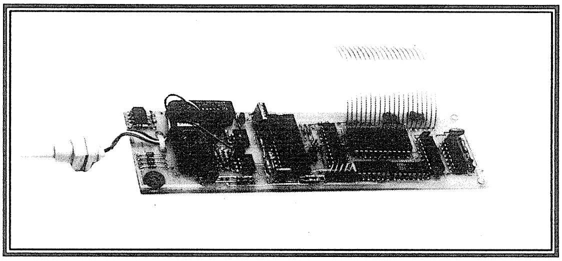
```
VIEW PRINTER DRIVERS
Function                          HT Codes      Printer Codes

Printer                                         RX80/FT
Answer file                                     A.PD1
Printer driver program            PD1           PD1
Type                                            Dot mat
Initialisation code                             ESC "@"
Auto line feed                                  Yes
Italics - enter                   *-*           ESC "4"
        - leave                   *-*           ESC "5"
Alternative font
        - enter
        - leave
 French - enter
- normal re-enter
Emphasized - enter                *--           ESC "E"
           - leave                *--           ESC "F"
Double strike - enter             ***           ESC "G"
              - leave             ***           ESC "H"
Underline - enter                 -             ESC "-" 1
          - leave                 -             ESC "-" 0
Underline spaces?                               Yes
Code 8 back space                               Yes
Back space - enter
Super/subscripts?                               Yes
Superscript - enter               **            ESC "S" 0
Subscript - enter                 *-            ESC "S" 1
Normal - enter                    **-           ESC "T"
Code for $ - enter                              "S"
Code for # - enter                              "#"
Code for # - enter                              35
Extended character set                          No
Exact space (pad character)                     Yes (@)
Reset all functions off
use at start of document          *--- *---
```

Figure 5. Sample View Printer Driver Codes

Some Novel Uses for the Master Modem

David Pilling, author of the BEEBUG Command ROM, explains how to exploit some of its special features in the version supplied with the BEEBUG Master Modem.

The BEEBUG Master Modem, in conjunction with the special version of the Command ROM, provides a compact and efficient way of data communications with other computers. However, the novel use of the Master's sound system by this modem means that it can also be used for speech communication. Indeed, it is possible to set up a complete answerphone system, as we shall see.



The listing presented here, contains four useful procedures for use with the Master Modem. These are:

PROCconnect
This causes the modem to seize the phone line i.e. go 'off hook'. So this procedure is equivalent to you picking up your telephone handset.

PROCdisconnect
This makes the modem drop the line i.e. go 'on hook'. Which is just what happens when you replace your telephone's handset.

PROCanswer
This procedure waits until the telephone rings. It then uses PROCconnect to 'pick up' the phone.

PROCcall(A$)
This dials the number given in the string A$ and leaves the telephone off hook.

You might think that these procedures duplicate some of the star commands in the Command ROM. However, this is not so because the star commands, either generate a carrier tone or look for one. The procedures above simply connect the Master's sound system to the phone line.

To show how useful these procedures are, the rest of the listing turns your Master into an answering machine. A few add-ons are needed though. First, the *SAY commands are intended to send text to a speech synthesizer. I used Superior Software's SPEECH! package, but others will no doubt do if they use the internal sound system. Secondly, the *MOTOR commands in the listing are supposed to switch on and off a cassette tape recorder whose microphone is placed next to the Masters loud speaker. Notice that this *SAY clashes with that in Command, so you must disable the latter before using the program.

A few other ideas spring to mind, you could use just the speech synthesizer to give a message to callers. Obviously, you could use TIME$ to implement your own 'speaking clock' service or using the procedure PROCcall start your own alarm call business. More practically, if you leave your Master controlling experiments or equipment overnight, you could easily make it call you to raise the alarm if anything went wrong, and tell you of the problem. Similarly, if you leave your computer to look after your house, you could call it and be given a status report.

To take things much further than this, one really needs a way of sending messages back to the computer. Since speech recognition is still impractical, one would go for a DTMF (Dual Tone Multi-Frequency) decoder, which could decode the tones from 'touch tone' phones. Such computer systems with a speech synthesizer controlled by a tone keypad are now fairly common with banks etc. However, so far a DTMF decoder has yet to appear for the Beeb. Incidentally, you can use the Master Modem to tone dial; simply use the *TONE ON command and then dial in the usual way.

Note that there may be insufficient volume on some combinations of hardware and software. Unfortunately, this is something which can only be determined by trying the program on different systems.

```
   10 REM Program ANSWER
   20 REM Version B1.0
   30 REM Author  David Pilling
   40 REM BEEBUG  November 1988
   50 REM Program subject to copyright
   60 :
  100 REPEAT
  110 PROCanswer
  120 *SAY"I'M SORRY WE'RE NOT HOME
RIGHT NOW"
  130 *SAY"BUT IF YOU LEAVE YOUR NAME
AND NUMBER"
  140 *SAY"WE'LL GET BACK TO YOU AS
SOON AS WE CAN"
  150 *SAY"PLEASE LEAVE YOUR MESSAGE
AFTER THE TONE"
  160 :
  170 *MOTOR 1
  180 VDU 7
  190 I%=INKEY(6000)
  200 PROCdisconnect
  210 *MOTOR 0
  220 :
  230 UNTIL FALSE
  240 END
  250 :
 1000 DEF PROCconnect
 1010 ?&FE81=&3B
 1020 ENDPROC
 1030 :
 1040 DEF PROCdisconnect
 1050 ?&FE81=0
 1060 ENDPROC
 1070 :
 1080 DEF PROCcall(A$)
 1090 ?&FE82=3:?&FE82=&56:?&FE81=&19
 1100 *FX15
 1110 I%=INKEY(200)
 1120 FOR J%=1 TO LEN A$
 1130 N%=VAL(MID$(A$,J%,1))
 1140 IF N%=0 N%=10
 1150 FOR L%=1 TO N%
 1160 PROCdisconnect
 1170 I%=INKEY(6)
 1180 PROCconnect
 1190 I%=INKEY(4)
 1200 NEXT
 1210 I%=INKEY(70)
 1220 NEXT
 1230 ENDPROC
 1240 :
 1250 DEF PROCanswer
 1260 ?&FE82=3:?&FE82=&42
 1270 ?&FE81=0:X%=0
 1280 T%=TIME
 1290 REPEAT
 1300 IF TIME-T%>1000 X%=0
 1310 A%=?&FE82:UNTIL(A% AND 4)=4
 1320 A%=?&FE82:A%=?&FE83
 1330 PRINT"RING"
 1340 A%=INKEY(55):X%=X%+1:IFX%<4 GOTO
1280
 1350 PROCconnect
 1360 ENDPROC
```

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

MATRICES IN BASIC (Vol.7 No.3&4)

Line 3140 of Listing 1 from the first part of this article was printed incorrectly. It should read:

```
3140 STA arname:JSR findar
```

In part 2 of this article, the changes given for Basic I users on page 26 were incorrect. The second line should read:

```
10182 divmfa=&A6B8
```

File Handling For All (Part 6)

by David Spencer and Mike Williams

This month will introduce some new Basic keywords related to file handling, and recap on the old ones.

BGET AND BPUT

All our file handling up to now has involved the use of PRINT# to send data to a file and INPUT# to read it back again. We have already said that Basic stores values treated in this way in its own special format. We have also said that integers take five bytes in a file, reals six bytes, and strings take one byte for each character plus an extra two.

This use of Basic's own data format makes the writing of file handling routines fairly straightforward, though it is not without drawbacks. One problem is that each value might take up far more space than is actually needed. For example, consider the case of writing many single letters to a file as strings. Each letter will take up three bytes in the file, although the actual character can be represented in just one byte. Another drawback is that we are limited by the restrictions that Basic places on us. This means, for example, that strings can be no longer than 255 characters, and integers cannot be greater than $2*10^9$.

All these problems can be solved by bypassing the Basic formats, and accessing the file byte by byte. Basic provides us with two keywords to do just this, BPUT and BGET. As their names suggest, BPUT puts a byte out to a file, and BGET gets a byte back again.

Before expanding on these operations we need to clarify just how much information a byte can hold. Essentially, one single byte can hold any integer value between zero and 255. The number stored in a byte can be thought of as a character code, in which case one byte can hold a single character. In order to store more complex values, for example larger integers, real numbers or character strings, we need to use more than one byte. We will look at how this is done shortly, but first the commands themselves.

BPUT is used in the form:

```
  BPUT#F,data
```

which will put the value of *data* to the file whose handle is *F* as a single byte. The value of *data* should be an integer between zero and 255. Basic will actually take the integer value of the argument and AND it with 255 before sending it to the file. This means that 39.3 would be sent as 39, and 260 would be sent as 4. To send a character using BPUT, it must first be converted to the corresponding ASCII character code using ASC. For example, to send the letter 'A' to a file you could use:

```
  BPUT#F,ASC"A"
```

ASC can of course take a string as its argument. If the string is longer than one character, then only the first character is used.

BGET performs the reverse of the BPUT operation. To read the value of a byte from a file into the variable *data*, you could use:

```
  data=BGET#F
```

The value read will always be an integer between 0 and 255. To convert the value into a character, CHR$ can be used. For example:

```
  data$=CHR$(BGET#F)
```

will set *data$* to the single character whose ASCII code was read from the file.

BGET and BPUT are very powerful commands, because they allow access to a file at the lowest possible level. Unfortunately, as is nearly always the case, the power of these commands means that it is very easy to make mistakes and end up in total confusion. The key to writing a successful file handling program is to design the record and file layout right at the start. The first thing to do is to decide what each field of each record must represent, and then work out how many bytes will be needed to accommodate this.

The ways of representing different types of data are quite arbitrary, but we shall look at some examples here. The simplest case is where all the data will be integers between zero and 255. In this case, each value can be represented by a single byte. A slightly more complex case is when the numbers are still integers, but are greater than 255. In this case we need to use the Basic operators DIV and MOD to split up the number into byte sized chunks. If we allow two bytes for each number, then integers in the range 0 to 65535 can be represented. The way to write such a number to a file is to use:

```
BPUT#F,value MOD 256
BPUT#F,value DIV 256
```

which writes the number stored in *value* as two bytes, with the low byte first. The corresponding code to read the value back in is:

```
value=BGET#F+256*BGET#F
```

Storing real values byte by byte to a file is a lot more involved. Basic uses a five byte format to store real numbers, but the conversion between this and the actual value is quite complicated, and beyond the scope of this series. In general, we would not recommend that you attempt to handle real values using BPUT and BGET. This might sound a bit restrictive, but you should stop and consider if you really need to use reals anyway. One case where a real value might be used is to store a price in pounds, for example £23.49. However, it is just as easy to store the value in pence, in this case 2349p, and then only integers are needed.

You will probably find that in almost all cases you can do away with the need to store real values in files. You might argue that this results in additional processing, but this argument falls down when you consider that the handling of integers is much faster than that of real values. Another possibility already mentioned earlier in the series is to convert real values to strings using STR$ when writing them to a file, and convert them back with VAL when they are read back again.

There are many different conventions by which a string of characters can be written to a file on a byte by byte basis. You might recall from earlier in the series that Basic stores a marker byte followed by a byte containing the length of the string, and then the characters of the string in reverse order. Another way of storing a string would be to store just the characters, and terminate the string with a special value. Typically, this terminator would be either a zero byte or a carriage return (ASCII code 13). It is not necessary using this technique to store the length of the string. Writing a string in this way could be performed thus, assuming that *data$* is the string to be written:

```
FOR count = 1 TO LEN data$
   BPUT#F,ASC(MID$(data$,count,1))
NEXT
BPUT#F,13
```

This outputs each character in turn, and then writes a carriage return as a terminator.

To read the string back in, the following could be used:

```
data$=""
REPEAT
   data=BGET#F
   IF data<>13 THEN data$=data$+CHR$data
UNTIL data=13
```

This builds up the string *data$* with bytes from the file until the terminating carriage return is reached.

If all our strings are of a fixed length, then even a terminator isn't necessary. Instead, the correct number of characters can just be written or read. For example, the following piece of code would write *data$* to a file in a twenty character field. If the string is less than twenty characters it is padded with spaces.

```
FOR count = 1 TO 20
   IF count > LEN(data$) THEN BPUT#F,32
   ELSE BPUT#F,ASC(MID$(data$,count,1))
NEXT
```

The same string could be read back in using:

```
data$=""
FOR count = 1 TO 20
   data$=data$+CHR$(BGET#F)
NEXT
```

The case of strings longer than 255 characters is more complex, because Basic doesn't allow these to be stored in memory as strings. We therefore need to develop our own routines to enter long strings, store them to a file, read them back again, and finally print them. The way to do this is to claim a block of memory using a variant of the DIM command, and store the characters that make up the string in this area. The following routine claims some memory and reads a string from the keyboard.

```
DIM string 1000
pos=0
REPEAT
   A=GET
   IF A=127 THEN pos=pos-1
   VDU A
   string?pos=A
   pos=pos+1
UNTIL A=13
```

This routine reads characters into the string until carriage return is pressed. The string is stored in the memory locations starting at *string*, and is terminated with a carriage return. The delete key can be used at any point to remove the last character entered, but the use of Ctrl-U to delete the whole line is not implemented. Obviously this routine could be expanded in many ways.

Writing a string stored in this format to a file could be achieved using:

```
pos=-1
REPEAT
   pos=pos+1
   BPUT#F,string?pos
UNTIL string?pos=13
```

This merely writes the bytes until the terminating carriage return is encountered. To write a string as a fixed number of characters is slightly more difficult because the LEN function can't be used to find the length of the actual string. A suitable routine to write a space-padded string 1000 characters long could be:

```
pos=-1
flag=FALSE
FOR count = 1 TO 1000
   pos=pos+1
   char=string?pos
   IF char=13 THEN flag=TRUE
   IF flag THEN BPUT#F,32
      ELSE BPUT#F,char
NEXT
```

The string could be read back using:

```
pos=0
REPEAT
   char=BGET#F
   string?pos=char
   pos=pos+1
UNTIL char=13
```

for the string terminated by a carriage return, or:

```
FOR pos = 0 TO 999
   string?pos=BGET#F
NEXT
```

for the fixed length version.

Finally, printing the string is accomplished with:

```
pos=0
REPEAT
   char=string?pos
   VDU char
   pos=pos+1
UNTIL char=13
```

or the equivalent for the fixed length string.

FILE POINTERS REVISITED

Throughout the series we have mentioned the use of PTR# without explaining exactly how it functions.

Whenever a file is opened it is assigned a pointer which can point at any byte within the file. As the file is opened its pointer is set to the value zero which points at the first byte of the file. Normally, each time a byte is transferred to or from the file, the value of the pointer is incremented by one. For read only files the pointer stops being incremented at the end of a file, and an end of file condition is returned. For writable files the file is extended as necessary to allow the pointer to be incremented.

Random access to files is accomplished by directly manipulating the file pointer using the keyword PTR#. This can be used in two forms: as a function to read the current value of the pointer; and as a statement to set the pointer to a new value. For example:

```
pos=PTR#F
```

will set *pos* to the current value of the file pointer for channel *F*, while:

```
PTR#F=newpos
```

sets the file pointer for channel *F* to the value in *newpos*. If this would set the pointer to past the end of a file then an error is returned for a read only file, otherwise, the file is extended just enough to include the byte pointed to by the new pointer value.

So far in this series we have shown how you can use PTR# to jump to an absolute position in a file. This means that, for example, if you know the length in bytes of each record, and the length of the File Description Record (FDR), you can go straight to the start of any record.

Another way to use PTR# is to move the file pointer relative to its current position using:

```
PTR#F=PTR#F+offset
```

This will move the pointer by the value of *offset*. If this is positive then the pointer will be moved further on in the file, while a negative value will move the pointer backwards.

There are several cases where this relative movement could be useful. For example, suppose that a database program had a menu option to go back to re-load the current record being edited. This could be used if the contents of that record had been messed up accidentally while trying to edit them. Re-loading the record would revert to the version saved in the file. Assuming that the current record had just been read in, then the file pointer would be pointing to the start of the next record in the file.

To re-load the current record it is necessary to move the file pointer back one record, so that it points at the current one, and then call the routine to read a record from file. The simplest way to move back a record would be to use:

```
PTR#F=PTR#F-recordlen
```

where *recordlen* is the length (in bytes) of one record. This will work in all cases, regardless of where in the file the record is.

EOF AND EXT

We have already met the end of file function (EOF). This can be used to tell whether the file pointer is at the end of the file or not. The statement:

```
flag = EOF#F
```

will set *flag* to TRUE if the pointer for file *F* is at the end of the file, or to FALSE otherwise.

Using EOF is a very simple way of telling if all the data has been read from a file or not. For example, a simple program to print all the records from a data file might be structured:

```
open data file
REPEAT
   read record
   print record
UNTIL EOF#file
close file
```

In our more sophisticated file structure using a File Description Record (FDR) to indicate the number of records, it should not be necessary to resort to testing for the end of file condition. However, there are cases where the EOF function is useful. For example when copying the entire contents of one file into another. EOF can be used to tell when the whole file has been copied.

EOF can also be used with output files, although it is not nearly as useful as it is when reading files. The easiest way to think of EOF for an output file is to say that if EOF is true, then writing any more data to that file will extend it.

A file function which we have not met so far is EXT. This is short for extent, and is used to read the length of a file. For example:

```
length=EXT#F
```

will set *length* to contain the length of the file whose handle is *F*. The main use of EXT is to find out the length of a file that has been opened for input. This could in turn be used to find the number of records stored in the file.

On the Master 128, Compact or Archimedes it is also possible to change the length of a file using:

```
EXT#F=newlength
```

which is mainly of use in reducing the length of a file. On earlier machines, neither the filing systems or Basic contained the necessary code to do this, and therefore this use of EXT should be avoided.

Next month we will move on to more elaborate ways of organising files which will make accessing records easier.

David Spencer takes a look at tree structures.

If you have been following the last few Workshops, you should by now be familiar with pointers, and how they can be used to form linked lists. Just to recap, a linked list is a series of data records with each one containing a pointer to the next record. Such a structure is essentially one-dimensional, and this leads to the alternative name of a 'linear list'. This month we will broaden our horizons and look at what happens if each record points to more than one other. Such a data structure will no longer be linear, but is instead two-dimensional.

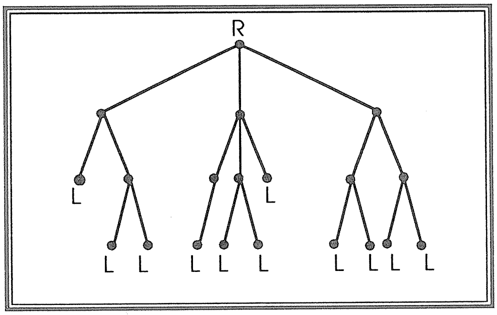


Figure 1

ROOTS, BRANCHES AND LEAVES

Figure 1 shows how our new data structure could look if drawn on paper. The 'blobs' represent the data records, or nodes as we shall call them, and the arrows indicate which nodes a particular record points to.

The node at the top of figure 1 (labelled 'R') is special in that it has no other nodes pointing to it. This node is therefore the starting point and forms a 'foundation' for all the other nodes. For reasons that will become clear this is called the root node. The root node contains all the fields needed to store its data record, and additionally, three pointer fields. These fields point to the three nodes shown immediately below the root in figure 1. This process continues, so each of these three nodes points to other nodes and they then point to others and so on.

If you turn the magazine upside down at this stage you should be able to see why we refer to the starting point as a *root* node. Our first record acts as a root for all the others, which then grow out of it. The more nodes that are added, then the more the diagram looks like a tree (albeit upside down), and therefore this type of data structure is called a *tree.* Continuing the botanical theme, the lines representing the pointers are referred to as *branches,* and the end nodes, that is records that point to no others, are called *leaves.* These are marked with the letter 'L' in figure 1.

If you take another look at figure 1 you will see that the number of branches from any particular node varies. The root node has three branches; the leaves have no branches (that is their definition); and the other nodes have between one and three branches. In general, any node of a tree can have any number of branches, and this number doesn't need to be the same for all the nodes in a tree. The more branches a tree has, the more 'bushy' it will appear when drawn as in figure 1. Another property of a tree is its depth. The depth of a node is defined as the number of branches that must be followed to reach that node. Thus, the depth of the root node is zero, the depth of any nodes pointed to by the root is one, and so on. The depth of a tree is simply the depth of the deepest node in the tree. For figure 1 the depth is three.

A final important point about trees is that at no time does a node point to another at a higher level. In other words, the branches always go *away* from the root, not towards it. This is very important, because as we will see, some of the routines used to read and write records within a tree will not function if any branches go upwards.

CREATING AND ACCESSING TREES

The routines necessary to add records to a tree, and then to read them back again, are very similar in principle to those used for linked lists.

Referring back to figure 1, you should be able to see that if you take any node, then that node can be considered as the root of a mini-tree, or *sub-tree*. If you have been following the last few workshops then this should immediately cause the word 'Recursion' to spring to mind.

Recursion is the key to accessing trees. As an example, suppose that you wanted to print out the records from all the nodes within a tree. The easiest way to do this is to write a procedure that prints out the contents of a given node, and then calls itself recursively for each of the sub-trees pointed to by its pointers. If you follow the progress of such a procedure you should be able to see that by starting at the root, all the nodes in the tree are visited. Figure 2 shows a tree with each node numbered according to the order in which it is visited.

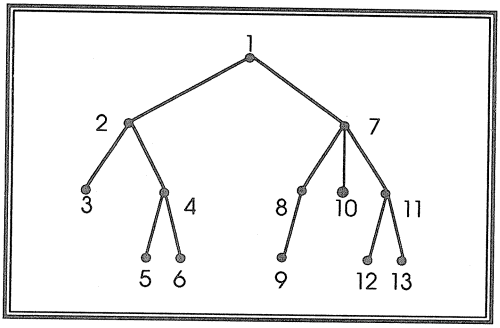


Figure 2

THE BINARY SEQUENCE SEARCH TREE

Rather than continue to explain abstract theory, I will give a practical example of one particular type of tree. This is the Binary Sequence Search Tree (BSST), which is a clever way of sorting a set of records into order.

For our example we will assume that we want to read all the words contained within a text file and then print out a list of all the unique words together with the number of times each has occurred. Furthermore, this list must be in alphabetical order. A program to do just this using linked lists was given in the Workshop in BEEBUG Vol.7 No.4. However, while that program was an improvement on the use of arrays or the like, it is still fairly slow and inefficient.

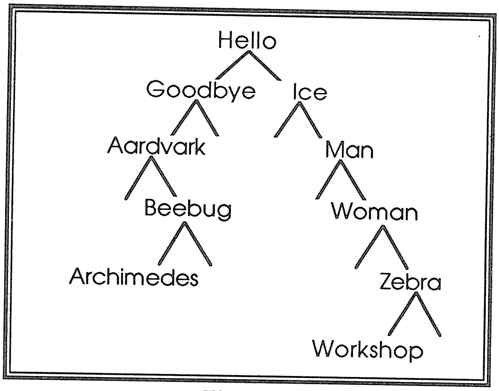


Figure 3

The main reason for this is that the program had to search through the entire list of words until the correct place for the new one was found. When there are many words in the list, and the new one starts with 'Z', this can be quite time-consuming.

The criterion we used to put words in the linked list was very simple: All the words following a particular one were those which came later in the alphabet. This meant that the first word in the list came first alphabetically, the last word came last, and all those in between were in the right order.

Now consider a change of rules. This time each record containing a word points not to one, but to two other nodes. This immediately leads to a tree structure, with each node having at most two sub-trees attached to it. Such a tree is known as a *binary tree*. With two pointers, our rule about how the records are put in order no longer applies. Instead, we will say that for any word from any record, all *the words in the left sub-tree* of that record come before that word, and all those in the right sub-tree come after it. If this isn't clear, look at figure 3 which shows such a tree containing just ten words.

Adding words to our BSST is really very simple. The process is best summarised by a series of steps:

1) Compare the new word with that stored in the root node.

2) If the new word is less than that at the current node, then examine the left-hand branch, otherwise look at the right-hand branch.

3) If there is no branch to take then create a new node and link it in.

4) Otherwise, repeat the whole process treating the sub-tree as if it was a complete tree.

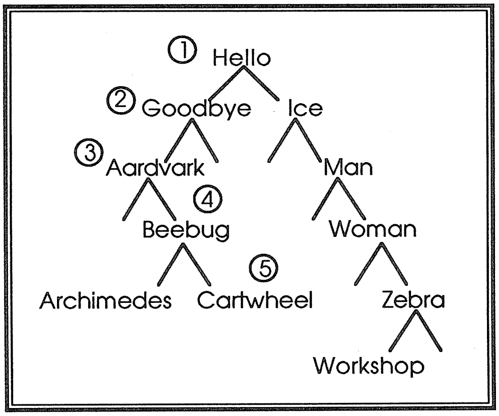


Figure 4

Figure 4 shows the steps involved in inserting the word 'CARTWHEEL' into the BSST of figure 3.

Printing the contents of a BSST is even more simple. All that is needed is a routine that will call itself recursively to print the contents of the left sub-tree of a node, then print the contents of that node, and then print the right sub-tree recursively. If you start this process at the root node then the contents of the entire tree are printed out in order.

The program given below puts our BSST example into practice. As it stands, the program reads words one at a time from a text file, and at the end prints an ordered list together with the number of occurrences. The program will accept any plain text file as input. I strongly advise that you work your way through this program, because it will explain the manipulation of trees far better than any number of pages of text can do.

Next month we will look at one specialised application of a tree: namely to represent the moves in a game such as chess or draughts. We will show how the tree can be used to choose the best move to make.

```
   10 REM Program Word Count using BSST
   20 REM Version B 1.0
   30 REM Author  David Spencer
   40 REM BEEBUG  November 1988
   50 REM Program subject to copyright
   60 :
  100 DIM block 5000
  110 !block=0:free=block+4
  120 INPUT"Enter filename "file$
  130 X%=OPENIN file$
  140 ON ERROR CLOSE#X%:END
  150 IF X%=0 PRINT"File not found":END
  160 REPEAT word$=FNgetword(X%)
  170 IF word$<>"" PROCenterword(block,word$)
  180 UNTIL EOF#X%
  190 CLOSE #X%
  200 ON ERROR OFF
  210 total=0:words=0:PRINT
  220 PROCprintwords(!block)
  230 PRINT';total;" Words"';words;" Unique
words"
  240 END
  250 :
 1000 DEF PROCenterword(ptr,W$)
 1010 IF !ptr=0 THEN
!free=0:free!4=0:free!8=1:
$(free+12)=W$:!ptr=free:free=free+LEN
W$+13:ENDPROC
 1020 ptr=!ptr
 1030 IF W$=$(ptr+12) THEN ptr!8=ptr!8+1
:ENDPROC
 1040 IF W$<$(ptr+12) THEN PROCenterword
(ptr,W$) ELSE PROCenterword(ptr+4,W$)
 1050 ENDPROC
 1060 :
 1070 DEF PROCprintwords(ptr)
 1080 IF ptr=0 THEN ENDPROC
 1090 PROCprintwords(!ptr)
 1100 PRINT$(ptr+12);TAB(20);ptr!8
 1110 total=total+ptr!8
 1120 words=words+1
 1130 PROCprintwords(ptr!4)
 1140 ENDPROC
 1150 :
 1160 DEF FNgetword(file)
 1170 IF EOF#file THEN =""
 1180 REPEAT A%=BGET#file
 1190 UNTIL A%>64 OR EOF#file
 1200 A$=""
 1210 REPEAT A$=A$+CHR$(A% OR &20)
 1220 A%=BGET#file
 1230 UNTIL A%<65 OR EOF#file
 1240 =A$
```

Games Reviews

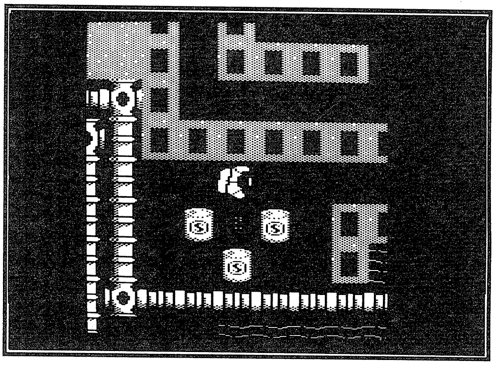
Six of the latest games releases reviewed by Lance Allison.

Over the last months BEEBUG has featured very few games reviews, possibly reflecting the more professional and serious use to which the BBC micros are applied nowadays. However, with Christmas just around the corner, I shall look at a number of the new games packages to have entered the market. If you need inspiration on what to get your son, daughter, or even husband(!) this Christmas look no further.

The first two games are *Pipeline* and *By Fair Means or Foul,* both from Superior Software. It is worth noting that Superior Software seem to have almost a monopoly on the BBC Games market at the moment. Not only do they market their own enormous range of software, but are also responsible for the Acornsoft name, marketing everything from Elite to Arcadians. One possible reason for their success is the excellent presentation of all their games both in packaging and in the well designed title screens and menus. They certainly have an excellent marketing division!

Pipeline is very much in the Repton style. The scenario places you on one of the moons of Jupiter, called Io. Due to Earth's enormous appetite for sulphur (for some reason), mining operations were established here. Unfortunately, the robot controlled extraction platforms are malfunctioning and you have been sent there to recover the precious drums of sulphur, close the platforms, and return to Earth.

In real terms this involves wandering around numerous rooms and walkways, avoiding nasties and collecting barrels of sulphur. The screens themselves are very attractive and the game is livened up by the presence of Pipelines down which you travel at great speed to re-appear elsewhere on the same level. The smooth scrolling and sensible use of colour make the game a pleasure to play. In the same manner as their other Repton type games, character and level designers are included, allowing new screens and characters to be created with a little time and care. Although I do not have the patience or inclination to design new screens myself, some people get a great deal of pleasure from doing so and this extends the life of the game.



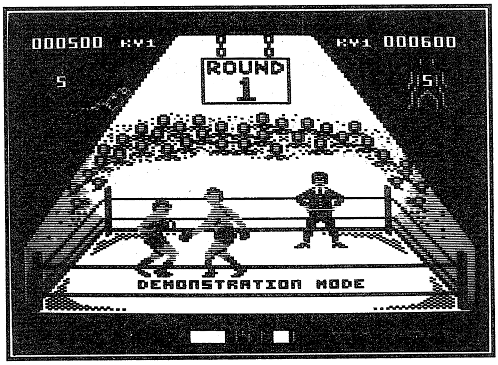
Pipline

In usual Superior Software tradition the packaging includes details of a competition. Besides several runner-up prizes, the player to have submitted the highest score by 31st January 1989 will receive a prize of one hundred pounds.

By fair means or foul is a good implementation of the 'boxing' style of game which has proved to be very popular recently. This game is beautifully presented, with very attractive title screens and demonstrations. Two players may box each other, or one player can box the computer. In either case, each player must select his character, and then enter his name. When you are placed in the ring fighting begins.

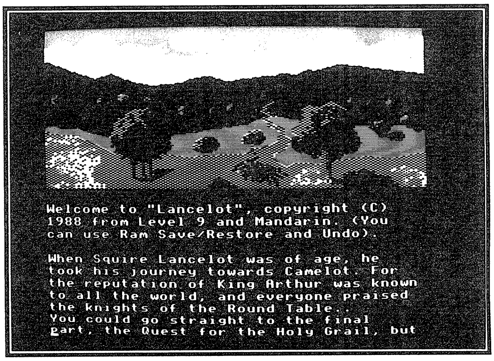
Mastering the controls is a skill unto itself. Fortunately there is a joystick option which helps immensely. Although it is quite fun to play against the computer, the game really comes into its own when the double player option is selected. As well as all the legal moves such as punch, guard low, and body blow, a number of illegal moves (illegal in the boxing world) such as the kick, groin punch, and head butt are also provided, but be careful that the referee does not see you using them!

All in all, the game is very entertaining, but I would recommend that you use a joystick to get the best from it (two joysticks of course for the two player version). Using the keyboard takes a lot of practice and, judging by the vigour with which the keys need to be hit, you may run the risk of damaging the keyboard. Again details of a competition are included. If you become a World Champion you qualify for a further draw, the winner of which will receive a fifty pounds cheque and a congratulatory certificate signed by Barry McGuigan.



By Fair Means of Foul

Moving away from the fast moving arcade games, Level 9 has recently released a new adventure game entitled *'Lancelot'*. In the past, Level 9 has produced some excellent adventures and I have the greatest of respect for the company, indeed this latest adventure has to be the best yet. Only Level 9 could integrate a massive text adventure with completely stunning graphics. The first thing to note is that the adventure will not run on a standard BBC model B, you must have a bank of sideways RAM or some shadow RAM. To get the most from the game you will need both shadow and sideways RAM. However, if you have a Master or a B+ 128 you will have this memory already.



```
Welcome to "Lancelot", copyright (C)
1988 from Level 9 and Mandarin. (You
can use Ram Save/Restore and Undo).

When Squire Lancelot was of age, he
took his journey towards Camelot. For
the reputation of King Arthur was known
to all the world, and everyone praised
the knights of the Round Table..
You could go straight to the final
part, the Quest for the Holy Grail, but
```

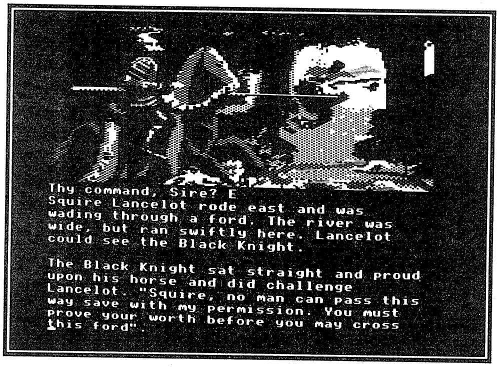
Lancelot - initial screen

The packaging is very impressive incorporating the disc, a map, and a very expensive looking manual called *Lancelot - Setting the Scene.* This manual gives a detailed guide to new adventurers, and sets the scene for already established adventuring freaks.

Getting down to the adventure itself, you play the part of Sir Lancelot on his quest for the Holy Grail. Being a great fan of this legend and knowing the story quite well made the game that much more enjoyable. Having said that, I have not made as good progress as I would have wished.

As soon as you start to play, you will notice that the game understands practically any English statement that you might wish to enter. This is down to the command line interpreter, for which Level 9 is renowned. Despite its extensive vocabulary, the software responds reasonably quickly on the eighty column screen.

As I mentioned earlier, the game also incorporates much graphics. The graphics take up just over half the screen but may be moved up and down so as to view more, or less, text. This is an excellent system, and works very well. The display is changed after every few commands, for a new and equally impressive colour screen.



```
Thy command, Sire? E
Squire Lancelot rode east and was
wading through a ford. The river was
wide, but ran swiftly here. Lancelot
could see the Black Knight.

The Black Knight sat straight and proud
upon his horse and did challenge
Lancelot. "Squire, no man can pass this
way save with my permission. You must
prove your worth before you may cross
this ford".
```

Lancelot - the Knight

This adventure also incorporates a competition to win a 'Holy Grail' for yourself (estimated value is about five thousand pounds!). Finally, a hint sheet is available from Level 9 if you really need it.

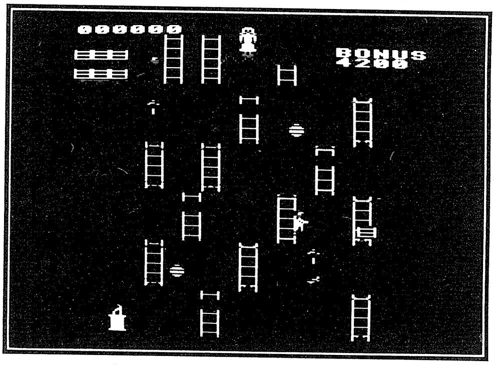
Next are two new games compilation discs recently released by Superior Software. Both of these packages contain four well-established games, and offer superb value for money. The fist compilation disc, ***Play It Again Sam 3***, contains *Commando, Palace of Magic, Killer Gorilla,* and the sequel *Killer Gorilla* 2.

Commando is a simple game that involves guiding a soldier around a field whilst under enemy gunfire. Unfortunately the game appears to have been written rather badly, hence the bullets move rather slowly and the graphics are far from smooth. However, the game is entertaining if rather outdated.

Palace of Magic is a 'Manic Miner' type of game in which you guide your man around countless screens avoiding hazards, and collecting objects to solve puzzles. The graphics are really quite smooth and it is the sort of game that quickly becomes addictive.

Killer Gorilla and the sequel are both highly entertaining reproductions of the famous arcade game Crazy Kong. Crazy Kong has kidnapped your girlfriend and has taken her to the top of a half built building. Your task is simply to climb the framework and rescue her. However, there are many obstacles to avoid and objects to collect.

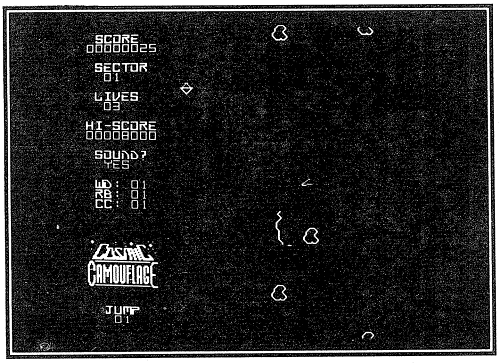
The sequel is based around a similar scenario, but the game is sufficiently different to warrant being sold on the same disc. Although Killer Gorilla was one of the first good games to reach the BBC market, it still holds up well against newer games, and there should be a copy in any decent games collection.



Killer Gorilla

The second compilation disc, ***Play It Again Sam 4*** (as if you hadn't guessed!), contains *FRAK!, Spellbinder, Cosmic Camouflage,* and the *Grand Prix Construction Set. FRAK!* is another 'classic' which had a cult following three or four years ago. The idea is to guide Trogg the cave man from one level to the next by climbing ropes, jumping from platform to platform, and killing monsters by throwing your yoyo at them. The graphics and (especially) the sound effects still make this game great entertainment.

Cosmic Camouflage is simply a version of the rather old 'Asteroids' type game. Manoeuvre your spacecraft around the screen, blowing up asteroids and mysterious flying saucers. The graphics are well done, but I have seen better implementations on the market.



Cosmic Camouflage

Spellbinder is a three dimensional graphics adventure game incorporating many tricks and puzzles. Guide the Wizard from room to room collecting spells, and ridding the place of its' evil inhabitants by magical means.

The Grand Prix Construction Set is a rough implementation of the popular arcade game Pole Position, except that you can actually design your own race course. Another nice feature is the two player option. In this mode, a separate view of the course is displayed for each driver. This type of racing game is well suited to younger children because the controls are easy to master. However, this is not a racing car simulation, and does not compete with the likes of Revs, originally written by Acornsoft.

That about covers the new releases known so far, though Superior at least is rumoured to have a major games release imminent. However, Impact Software has recently released two discs called *Cheat it Again Joe Volume One and Volume Two.* Each disc contains cheats for twenty popular games. There is not enough room to list all of the games covered but, looking through the list, they include everything from Elite to Starship Command. This may well be the perfect gift for the games player that has everything (well almost!).

SUMMARY

If you are looking to purchase games as a present for someone else, the two compilation discs offer excellent value for money. There is very little between them, so I leave the choice to you. For the games enthusiast 'Pipeline' and 'By Fair Means or Foul' offer new entertainment and are worth purchasing. As I said previously, Lancelot from Level 9 is in a class of its own. No BBC owner should be without a copy in their Christmas stocking!

PRODUCT DETAILS

Product	**Lancelot**
Supplier	**Level 9, Europa House, Adlington Park, Adlington, Macclesfield SK10 4NP. Tel. (0625) 878888**
Price	**£14.95 inc VAT (5.25" disc)**

Product	**Cheat It Again Joe Volumes 1 And 2**
Supplier	**Impact Software, Neepsend House, 1 Percy Street, Sheffield S3 8AU. Tel.(0742) 769950**
Price	**£11.95 inc VAT each (5.25" disc)**

Products	**Pipeline By Fair Means Or Foul Play It Again Sam 3 and 4**
Supplier	**Superior Software, Regent House, Skinner Lane, Leeds LS7 1AX. Tel.(0532) 459453**
Price	**£11.95 inc VAT (5.25" disc) £14.95 inc VAT (3.5" disc £9.95 inc VAT (tape)**

1st course

Calling the Operator

This month, in the First Course spot, Mike Williams explains how to call operating system routines from within Basic.

The Beeb's operating system is itself just a program, albeit rather more complex than that written by the average user. It is in machine code, and unlike most programs is permanently installed in your machine so that it is immediately available as soon as you switch your computer on.

The operating system, like any self respecting program, consists of separate sub-routines, and many of these are documented within the User Guide. Furthermore, these routines may be easily called from within any Basic program. Although many splendid programs can be written without once calling any of these routines, it is well worthwhile getting to know how to use them, and in some instances they provide the only means of performing certain functions.

CALL AND USR

Basic provides two separate ways of calling OS routines (indeed any machine code routines). These are the USR function and the CALL statement. In both cases the name (or address) of the appropriate routine must be given. USR is like a Basic function and thus returns a value. CALL is more like a procedure call and no value is returned. In both cases, the processor's A, X and Y registers are set to the values currently stored in A%, X%, Y% (least significant byte only) on entry.

With USR, the final contents of the four registers P, Y, X and A (one byte for each) are returned in a single 32 bit integer value. Let's look at an example. You are probably familiar with the *FX calls available in Basic. In fact, each *FX call accesses a routine in the operating system, and all these routines go under the general name of OSBYTE calls. The value passed in A% to the processor's accumulator determines which OSBYTE function is to be used. The OSBYTE routine itself is located at &FFF4 on all versions of the model B and Master series machines.

Some OSBYTE calls, principally those which return a value of some kind, are much better dealt with using USR. We will look here at OSBYTE with A%=&87. This reads the character at the current cursor position. On exit, the ASCII code of the character read is returned in the X register.

Thus we can write:

```
A%=&87
result=USR(&FFF4)
char=(result AND &FF00) DIV &100
PRINT char,CHR$(char)
```

These few lines will read the character at the cursor position, extract this as an ASCII value from the result returned by USR (by ANDing with &FF00 and shifting 4 bits right), and display both the code and the character. That's all there is to it in a simple case like this.

Often more than a single value must either be passed to the OS routine, or returned. In such cases, a small block of memory is used for this purpose, and as a result CALL is usually preferable to USR. For example, you are probably familiar with the use of the VDU19 instruction to assign any of the Beeb's 16 colours to a specified logical colour. Thus:

```
VDU19,2,4,0,0,0
```

would assign blue (colour 4) to logical colour 2. All subsequent references to colour 2 will invoke blue.

But suppose we wish to determine within a program the physical colour currently assigned to a given logical colour. There is no Basic keyword which will achieve this. However, there is an OS routine which will do just that. This particular routine is one of a number of such routines going under the general name of OSWORD. The value assigned to A% (as with OSBYTE), determines the particular OSWORD call needed, in this case &0B. The OSWORD routine itself is always located at &FFF1.

We have to specify the logical colour which we want to investigate, and the method of doing this is rather more complicated than might at first be expected. The two registers X and Y are set to the start address of five bytes somewhere in memory. As each register can only hold a single byte (maximum 8 bits), any 16 bit address has to be split into two parts. The lower 8 bits or 'low byte' go into the first register, and the top or 'high' byte goes into the second register.

This may sound complicated, but is easily achieved in Basic using DIV and MOD. For instance, if the five bytes are situated somewhere called *block* then we may write:

```
X%=block MOD 256
Y%=block DIV 256
```

This is a useful format to remember and can be used whenever a 16 bit address has to be split into its high and low bytes as here.

Thus our OSWORD call could be written as:

```
A%=&0B
X%=block MOD 256
Y%=block DIV 256
CALL &FFF1
```

The information returned by the OS call will be assigned to the variable *data*, and we will see how to unpack this in a moment. First a few more words about the location of the five bytes of memory referred to as *block*.

There are two ways of reserving any number of bytes in memory for use by a Basic program. The simplest is to use a variant of the DIM statement, for example:

```
DIM block 4
```

The DIM statement here reserves 5 bytes of memory which can henceforth be referenced by the name *block*. To use this OSWORD call in our example the first byte of the five is set to the logical colour number to be investigated:

```
?block=2
```

Note the use of the '?' indirection operator to assign the required value as the first byte in the data block. Then follows the call to OSWORD as already described.

The alternative approach is to use the 32 bytes of memory starting at location &70 which are specifically reserved by Basic for user programs. To use these bytes in our example we would write:

```
block=&70
```

in place of the DIM statement used before.

The final step in the process is to extract the information returned by this call. This has been assigned to the variable *data* in our example. In this instance the five bytes starting at *block* are assigned the five parameters specified in the VDU19 statement, viz 2,4,0,0,0. If we want to extract the reference to the physical colour we could then write:

```
colour=block?1
```

(note that ?block references the first byte of data, block?1 the second, block?2 the next and so on).

If such a function is required more than once in a program, then it is probably better written as a Basic function, except for the DIM statement which must be executed once only near the start of the program. Thus:

```
1000 DEF FNfind_colour(logical_colour)
1010 ?block=logical_colour
1020 A%=&0B
1030 X%=block MOD 256
1040 Y%=block DIV 256
1050 CALL &FFF1
1060 =block?1
```

Another example of the CALL statement is particularly useful to Basic I users, as it provides the only way of performing an OSCLI function. Basic II (and later) users can use OSCLI direct, but the CALL equivalent is worth knowing about anyway. The OSCLI function provides a way of executing any star command, but with more flexibility than is possible with the star command itself.

For example, a screen display can be saved (or loaded) using a star command. Thus:

```
    *SAVE Screen1 3000 8000
```

would save any mode 0, 1 or 2 screen under the name 'Screen1'. Similarly, writing:

```
    *LOAD Screen1 3000
```

would reload the screen display (note that the addresses used are always given as hexadecimal numbers, though without the usual '&'). Both commands assume that any shadow memory is switched off. When *SAVE and *LOAD are used in this way, the filename must be explicitly given. You cannot replace the filename with a suitable string variable to which a name has previously been assigned.

Now this could be quite useful if it were possible. You could write a program so that every time a predetermined key were pressed, the current screen was saved to disc but with a different name, say Screen1, Screen2, Screen3, and so on. We shall see how this can be fulfilled with the OSCLI function.

The OSCLI routine is located at &FFF7. The X and Y registers are set to the low and high bytes of an address (as before). This time we need to reserve more than five bytes of memory, and about 40 bytes will suffice for almost any star command we are likely to use in this way (so we must use the DIM method). This memory will be used to store the appropriate star command as a character string. Thus implementing the screen saving idea would take the following form:

```
    DIM os 40
    n%=1
    .........
    REPEAT
    .........
    $os="SAVE Screen"+STR$(n%)+" 3000
8000"
    X%=os MOD 256
    Y%=os DIV 256
    CALL &FFF7
    n%=n%+1
    ........
    UNTIL exit%
```

It is common practice to write the heart of an OSCLI call as a simple procedure.

```
1100 DEF PROCoscli(string$)
1110 $os=string$
1120 X%=os MOD 256
1130 Y%=os DIV 256
1140 CALL &FFF7
1150 ENDPROC
```

Our outline loop would then become:

```
    REPEAT
    ........
    PROCoscli("SAVE Screen"+STR$(n%)+" 30
00 8000")
    n%=n%+1
    ........
    UNTIL exit%
```

Such uses of *SAVE (and similarly *LOAD as well) would be quite impossible unless performed with the help of the OSCLI function. Notice, too, the very essential spaces (and screen addresses) included in the specification of the star command.

A simple routine to display each screen saved in turn could be written as:

```
    FOR i%=1 to s%
    PROCoscli("LOAD Screen"+STR$(i%)+" 30
00")
    G=GET
    NEXT i%
```

where s% determines the total number of screens to be displayed. The loop pauses after each new screen until the space bar (or any other key) is pressed.

Remember that the figures of 3000 and 8000 used in the the above examples on screen saving and loading assume a mode 0, 1 or 2 screen. Replace the 3000 by 5800 for modes 4 and 5, and 7C00 for mode 7. Users with Basic II or later can, of course, just use OSCLI with a star command, such as:

```
    OSCLI("LOAD Screen"+STR$(i%)+" 3000")
```

This completely replaces the PROCoscli procedure defined above.

To conclude this month's discussion we will look at a slightly more complicated example, this time a call to the disc filing system (DFS) rather than the OS. Like OSWORD, the OSFILE

Continued to page 68

Using Text Files as Data Files

Stuck for how to enter and edit the data you need for your programs? Barry Thorpe explains how your favourite word processor can come to the rescue.

From time to time I write programs to process different sorts of data from files. For example, I did my own analysis of the results of the last General Election. Since I want to be able to use the programs again, it is obviously necessary to have the means of creating and editing the data files to be processed. But to write special data entry and editing routines for each application is a chore. One solution is to use a database, assuming that the files it creates are comprehensible, and write one's own processing modules. An alternative, which I find more convenient, is to use a word processor. Provided that one keeps careful note of the order of fields in a record, and so forth, entry and editing couldn't be easier.

Since the file to be processed is a pure ASCII file and not a normal data file, routines are necessary to read the file and convert its contents to Basic strings. Once in the machine, the strings can be handled as one pleases. This method is not particularly speedy, but it is a very convenient way of handling data entry.

To set up a file, one simply types each field, numeric or string, followed by a Return. The end of a record can be marked by some such value as an asterisk, and the end of the file by, say, an "@". The routines shown below can be adapted for your own purposes. The function defined at line 1100 is the routine that pulls out a field, marked by a Return, from the text file. The procedure defined at 1000 shows how the function might be used in a program to read in a complete record.

An alternative to marking the end of a record with an asterisk would be to use a FOR-NEXT loop to pull in the fields, assuming that each record has the same number of fields. However, the end-of-record markers in the text file are useful reference points when editing. The method suggested can be used for non-standard length records in a file-header, for example, without further change.

Another possibility, especially if the records are short, is to type each record on one line with fields separated by slashes. In this case, the identifiers in the routines below should be changed from "field" to "record". With longer records, editing is less convenient using this method.

Outline Program to Read Data from a Text File

```
1000 DEF PROCprocess
1010 REM ......
1020 REM ......
1030 field$=FNgetfield(x)
1035 REM x is handle of data file
1040 REPEAT
1050 REM.......
1060 field$=FNgetfield(x)
1070 REM.......
1080 UNTIL field$="*"
1090 ENDPROC
1099 :
1100 DEF FNgetfield(ch)
1110 LOCAL t$,t,eofm
1120 eofm=ASC"@"
1121 REM Change to match your data
1122 REPEAT
1123 t=BGET#ch
1124 t$=t$+CHR$t
1125 UNTIL t=13 OR t=eofm OR EOF#ch
1130 t$=LEFT$(t$,LENt$-1)
1140 eof=(t=eofm OR EOF#ch)
1150 REM eof=TRUE if at end of file
1160 =t$
1170 :
```

ON SAVING THE TEXT FILE

Where the wordprocessor saves the text as a pure ASCII file, there is no problem. With Interword, set a zero left margin, turn paging OFF, and use option 8 - spool with NO codes. With Wordwise and/or View just omit all embedded commands and rulers from the text (although Wordwise can also spool its output). As an alternative use a text editor such as the Master's EDIT. This saves all text in pure ASCII form.

Using a word processor makes the creation and editing of data files very easy indeed, and this technique is used by the Route Planner program described elsewhere in this issue. B

Using Assembler (part 4)

This month Lee Calcraft looks at Events and Interrupts , and implements an Event-driven Clock.

Microprocessors such as the 6502, or even the ARM, can only execute one instruction at a time. But because of their speed of execution, it is possible to divide their time between a number of concurrent tasks. On sophisticated machines with very fast processors, it is possible for the operating system to divide the processor's time in such a way as to provide so-called multi-tasking. This is impractical on the 6502 because of its relatively slow speed, but by the use of interrupts and events, some of the features of multi-tasking at least can be achieved.

For example, if you have printer buffer software for your Beeb, you will see the computer perform two tasks completely independently of each other, and apparently simultaneously. This is achieved through the use of interrupts. At a more subtle level, interrupts are in operation continuously, allowing the Beeb's operating system to perform background tasks while the main task appears to continue uninterrupted. For example, when you press a key, the code for that key is inserted into the keyboard buffer whether you are running a program which uses keyboard input or not. Or to take another example, Basic's TIME variable is updated in response to interrupts caused by the system clock. This all happens as a background task without the user being aware that he has anything but the undivided attention of his processor, and it happens regardless of whether the user is running a Basic program or a piece of machine code.

Although it is quite feasible to intercept the Beeb's interrupt system to tag on pieces of user code, the machine's designers have provided an easier way of doing this through the implementation of so-called *events*. An event is essentially a pre-packaged interrupt which is readily available to the user. The operating system provides nine such events, and these are listed in table 1.

Event number	Cause of the Event
0	Output buffer becomes empty
1	Input buffer becomes full
2	Character enters input buffer
3	ADC conversion complete
4	Start of vertical sync
5	Interval timer crosses zero
6	Escape condition detected
7	RS423 error detected
8	Econet generated event
9	User event

Table 1. The ten events available

A particular event can be enabled by issuing the call *FX14,n, where n is the event number, and cancelled with *FX13,n. Once FX14 has been issued, every time that an event of the specified kind occurs, control is passed to the routine whose address is held at the event vector, located at &220 and &221 in RAM. In order to intercept a particular event enabled with *FX14,n, the programmer has simply to insert the address of his own routine at the event vector. At the end of his routine, which must take no more than 2 milliseconds to execute, he must pass control back to the operating system by jumping to the address which the event vector originally held.

AN EVENT-DRIVEN CLOCK

In order to demonstrate how events may be used, we will look at a complete example. The accompanying program implements an event-driven clock using event number 5 - triggered when the interval timer crosses zero. The interval timer is a special five-byte clock maintained in software by the operating system, and updated every 10 milliseconds under interrupts. If you run the program, you will see a digital clock appear at the top right-hand corner of the screen. Instructions also appear on screen for using the four function keys which control it. Key f0 allows you to set it to any time (e.g. entering 1509 will set it to 9 minutes past 3). Key f1 cancels the clock, f2 will restart it after it has been cleared, while f3 will zero it.

Because the clock is event-driven, it will continue to operate during almost any task which the computer performs, providing that locations &70 to &74 and &900 to &9FF remain undisturbed. The only thing which you cannot do is to perform cursor editing. If you want to do this, you will need to turn the clock off. This is because the clock uses legal text printing (rather than poking to the screen), and the Copy cursor (but not the ordinary text cursor) is disturbed. Because the routine does not poke to the screen, it will continue to work even after mode changes.

HOW IT WORKS

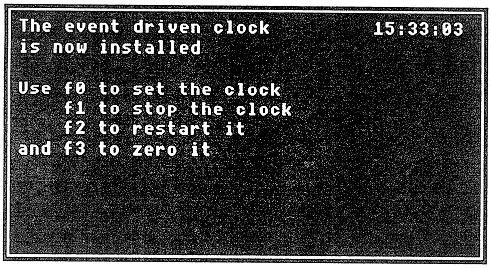
The assembler part of the program contains three separate routines called setup, *newcode* and *reset*. The first performs just two tasks. It saves the address held in the event vector (located at &220 and &221), and in its place it stores the low and high byte respectively of the address of the clock routine (i.e. it installs the address of *newcode*). It then performs an OSBYTE call, equivalent to *FX14,5. The third routine does exactly the opposite. It resets the event vector, and performs the OSBYTE equivalent of *FX13,5. Everything else is contained in the second of the three routines *newcode.*

The first task of *newcode* is to stack the 6502's registers. It is absolutely vital that any event (or interrupt) routine restores all registers, otherwise the foreground task that was interrupted by the event will almost definitely crash. Both status register and accumulator are stacked in line 550 of the program. Then the program checks the contents of the accumulator, which will hold the number of the triggering event. If it is not 5, then the routine is exited. This ensures that our routine only responds to events caused by the timer (event number 5). If we did not put this check in, then if a concurrent program (whether foreground or background) enabled another event, then our routine would incorrectly respond to it, as well as responding to the timer event.

Next we stack the X and Y registers, and initialise the clock with the OSWORD call at line 640. This places the value &FFFFFFFF9C into the interval timer's registers. The timer automatically counts upwards at the rate of 100Hz. After 100 pulses its registers will have gone from &FFFFFFFF9C to zero; and when this happens, event 5 is generated. By placing &FFFFFFFF9C into the timer every time that the event occurs, we achieve an event frequency of one per second. This is ideal for our clock, since there is no need to update the clock counters any more frequently than once a second.

The program uses three zero page locations (at &72, &73 and &74) to store the time in seconds, minutes and hours respectively. And to make the display routine simpler, we have used binary coded decimal representation, rather than straight binary. Thus while the number 19 would normally be held in a byte of memory as the hex number &13, in binary coded decimal it is held as &19. This allows us to hold the pairs of digits for seconds, minutes and hours in three single bytes of RAM, without giving us great problems when we come to extract decimal digits for the display. This is all achieved by placing the 6502 in so-called *decimal mode* using the SED instruction (i.e. SEt Decimal) at line 660. All time calculations are carried out in the code following this instruction, and ending at *countend*. By this point in the program, the three zero page locations holding the time have been correctly updated. The program then clears the decimal mode with the CLD instruction at line 990, and the clock is displayed using the subroutine *display* at 1080.

The display routine stacks the current cursor position, tabs to the correct position for the clock then, then calls subroutine *write* three times, with the accumulator containing hours, then minutes, and finally seconds. The user's cursor is then reinstated, stacked registers are unstacked, and the program hands back control to the Beeb's resident event handler.



```
The event driven clock        15:33:03
is now installed

Use f0 to set the clock
    f1 to stop the clock
    f2 to restart it
and f3 to zero it
```

Next month we return to a graphical theme, with a look at direct screen addressing.

```
   10 REM Event-driven Clock
   20 REM Version B 1.0E
   30 REM Author  Lee Calcraft
   40 REM BEEBUG  November 1988
   50 REM Program Subject to Copyright
   60 :
  100 MODE7
  110 PROCkeys
  120 PROCassem
  130 CLS
  140 PRINT"The event driven clock"
  150 PRINT"is now installed"'
  160 PRINT"Use f0 to set the clock"
  170 PRINT"    f1 to stop the clock"
  180 PRINT"    f2 to restart it"
  190 PRINT"and f3 to zero it"'
  200 CALL &900:REM initialise
  210 A%=5:CALL &903:REM start clock
  220 END
  230 :
  240 DEFPROCassem
  250 oswrch=&FFEE:osbyte=&FFF4
  260 osword=&FFF1
  270 vector=&220:REM event vector
  280 newvec=&70 :REM new vector addr
  290 secs=&72:mins=&73:hrs=&74
  300 ?secs=0:?mins=0:?hrs=0
  310 FOR pass=0 TO 1
  320 P%=&900
  330 [
  340 OPT pass*3
  350 .code
  360 JMP setup
  370 JMP newcode
  380 JMP reset
  390 \.........Initialise
  400 .setup
  410 \          Set vector
  420 LDA vector:STA newvec
  430 LDA vector+1:STA newvec+1
  440 LDA #newcode MOD &100
  450 STA vector
  460 LDA #newcode DIV &100
  470 STA vector+1
  480 LDX #5:LDA #14
  490 JSR osbyte
  500 RTS
  510 :
  520 \.........Main program
  530 \          executed each second
  540 .newcode  \
  550 PHP:PHA   \Stack status & acc
  560 CMP #5
  570 BNE notforme
  580 TXA:PHA   \Stack X register
  590 TYA:PHA   \Stack Y register
  600 :
  610 LDX #clockdata MOD 256
  620 LDY #clockdata DIV 256
  630 LDA #4
  640 JSR osword
  650 :
  660 SED
  670 \.........Seconds
  680 LDA secs
  690 CMP #&59
  700 BCS t1
  710 ADC #1
  720 STA secs
  730 JMP countend
  740 .t1
  750 LDA #0
  760 STA secs
  770 \.........Minutes
  780 LDA mins
  790 CMP #&59
  800 BCS t2
  810 ADC #1
  820 STA mins
```

```
830 JMP countend
840 .t2
850 LDA #0
860 STA mins
870 \.........Hours
880 LDA hrs
890 CMP #23
900 BCS t3
910 ADC #1
920 STA hrs
930 JMP countend
940 .t3
950 LDA #0
960 STA hrs
970 :
980 .countend
990 CLD
1000 JSR display
1010 PLA:TAY   \Unstack Y register
1020 PLA:TAX   \Unstack X register
1030 .notforme
1040 PLA:PLP   \Unstack status & acc
1050 JMP (newvec)\Resume OS routine
1060 :
1070 \.........Display Routine
1080 .display
1090 LDA #&86  \Read cursor posn
1100 JSR osbyte
1110 TYA:PHA   \Stack cursor posn
1120 TXA:PHA
1130 LDA #31   \Perform tab
1140 JSR oswrch
1150 LDA #30
1160 JSR oswrch
1170 LDA #0
1180 JSR oswrch
1190 \.........Display clock
1200 LDA #ASC" "
1210 JSR oswrch
1220 LDA hrs
1230 JSR write
1240 LDA #ASC":"
1250 JSR oswrch
1260 LDA mins
1270 JSR write
1280 LDA #ASC":"
1290 JSR oswrch
1300 LDA secs
1310 JSR write
1320 \.........Reinstate cursor
1330 LDA #31
1340 JSR oswrch
1350 PLA
1360 JSR oswrch
1370 PLA
1380 JSR oswrch
1390 RTS
1400 :
1410 \.........Write chr pair
1420 .write
1430 TAX
1440 AND #&F0
1450 LSR A:LSR A:LSR A:LSR A
1460 CLC
1470 ADC #48
1480 JSR oswrch
1490 TXA
1500 AND #&F
1510 CLC
1520 ADC #48
1530 JSR oswrch
1540 RTS
1550 :
1560 .reset    \Reset vector
1570 LDX #5:LDA #13
1580 JSR osbyte
1590 LDA newvec:STA vector
1600 LDA newvec+1:STA vector+1
1610 RTS
1620 :
1630 .clockdata
1640 EQUD &FFFFFF9C
1650 EQUB &FF
1660 ] NEXT
1670 ENDPROC
1680 :
1690 DEFPROCkeys
1700 *KEY0VDU21|M:VDU6:P."Set time":INP
UT"hhmm "A$:h=EVAL("&"+LEFT$(A$,2)):m=EV
AL("&"+RIGHT$(A$,2)):?&72=0:?&73=m:?&74=
h:CALL &906:CALL &900:A%=5:CALL &903|M
 1710 *KEY1 VDU21|MVDU6:P."Clock stopped
":CALL &906|M
 1720 *KEY2 VDU21|MVDU6:P."Clock restart
ed":CALL &906:CALL &900:A%=5:CALL &903|M
 1730 *KEY3VDU21|MVDU6:P."Clock zeroed":
?&72=&99:?&73=&99:?&74=&99|M
 1740 ENDPROC
```

The Comms Spot

Peter Rochford looks at the various protocols for transferring data.

In a previous Comms Spot I presented a guide to the methods of performing file transfers using your modem and comms software. This was in response to requests from many members who found the whole process difficult to get to grips with and needed to be able to exchange files with friends, or just download files from bulletin boards.

I did not discuss the various protocols available for file transfer in that article so this month we will take a look at these and their various merits.

ASCII

This is the simplest form of file transfer and allows the sending of 7 bit ASCII codes only. Control of information flow to and from each computer is effected by XON/XOFF flow control, achieved by the use of two control characters, Ctrl-S for XOFF and Ctrl-Q for XON.

When the input buffer of the receiving computer becomes full, it sends a Ctrl-S to the host to tell it to stop sending any more characters. This allows the receiving computer the time to perform the task of saving what it has in the buffer to disc, or to complete the dumping of it to a printer.

Once the buffer contents have been printed or saved, the receiving computer then resets the buffer pointer to the start of the buffer and sends a Ctrl-Q to the host to tell it to recommence transmitting. This process continues until all the information has been received and is performed entirely under automatic control of the software. You may, however, override the software and send either control character direct from the keyboard to halt and restart transmission.

ASCII file transfer should not be used unless really necessary, as it has numerous dis-advantages. To start with, you can only send 7 bit codes, so it is unsuitable for transferring Basic and machine code programs. Also, the host must inform the receiving computer in some way or other when it has finished sending all the information. Most importantly, there is no form of error checking. This means that your file may not arrive at the receiving end without suffering corruption by noise on the telephone line.

If you do have need to use ASCII transfer, then you will find that the majority of comms software provides the facility. It is usually referred to as *ASCII UPLOAD/DOWNLOAD* or as *TEXT UPLOAD/DOWNLOAD*. Also, many comms software packages allow all incoming text from a bulletin board, for example, to be spooled to disc. This is effectively downloading in ASCII.

XMODEM

This is the oldest and commonest file transfer protocol, allowing 8 bit files to be transferred, and incorporates error checking to preserve file integrity. The transmitted file is sent in 128 byte blocks, and these are checked by the receiving computer either by checksum, or by using the more reliable Cyclic Redundancy Check, to establish whether they have been corrupted.

If corruption has taken place, the receiving computer will send a request to the host to re-transmit that block. It will carry on doing this until the block is received correctly. The number of re-tries that will occur depends on the software you are using. Some will carry on ad infinitum asking the host to retransmit if the block is not received intact. The better software can be configured to stop after a certain number of re-tries and prompt the user, asking whether to abort the transfer.

Xmodem has several drawbacks. Firstly, only single files can be transferred one at a timemaking multi-file transfer sessions both time consuming and rather tedious. Secondly, neither file name nor file lengths are transmitted and these must be provided by the user for each of the transferred files.

YMODEM
This is a development of the original Xmodem protocol. The difference is that Ymodem uses a larger 1024 byte block length compared to the 128 byte length in Xmodem. The larger block length in Ymodem means that there is less time used up in waiting for blocks to be acknowledged. However, this is only advantageous when using lines which are relatively error-free.

There is also another Ymodem protocol called Ymodem Batch which again uses 1024 byte blocks but allows a number of files to be transferred at one time. It has the added advantage of maintaining file lengths and names.

KERMIT
This protocol (with the rather strange name) was originally developed to allow the transfer of files between micros and mainframes. Kermit is a batch protocol and therefore allows the sending and receiving of a number of files whilst maintaining both filenames and file lengths. Kermit can be used as a very straightforward method of file transfer in its basic form. It can, however, be very complex and sophisticated when a server option is implemented.

This server option allows access to the filing system of the host computer and means that you can perform all kinds of filing system operations. For example, you can move through directories of the host computer and catalogue directories enabling the selection and transfer of the files you require.

The amazing thing about Kermit is that it is public domain software. There are versions of it for most micros around and that includes the Beeb and the Archimedes. To get hold of it for yourself you can download it direct from the University of Lancaster.

CET
This is a protocol that is almost solely confined to viewdata systems such as Prestel. It is an error-checked protocol, like the others I have discussed, and maintains file lengths and filenames.

Those of you who have downloaded telesoftware from Prestel will have been using CET. It transfers 8 bit files but using 7 bit codes which are then interpreted by the receiving software and converted to their true 8 bit values. This explains why when displayed on the screen as they are received, they look nothing like the original program.

CONCLUSIONS
It is important to understand that none of the protocols I have discussed are compatible with one another. You must therefore, when attempting a file transfer, be using a protocol that is available on the host computer.

Most of the comms software around for the Beeb is equipped only with Xmodem. BEEBUG's own Command comms software features this. This is still the most popular and widespread of all the protocols, and you should have no trouble when accessing bulletin boards using this. Kermit, however, is fast gaining popularity in this country and is well worth getting hold of as it has numerous advantages over Xmodem.

If you own an Archimedes, you will be interested to know that full implementations of all the above protocols are available from within BEEBUG's Hearsay comms package, and it has the added advantage of allowing you to configure certain features of each protocol to suit your own needs.

Finally, it is worth mentioning that transferring large amounts of data via the telephone even at 2400 baud can be costly. There does exist a piece of software called Archive that compacts single or several files into one shorter file for sending much more quickly. When the compacted file is received, the same piece of software is used to 'unpack' the file and restore it to its original form. Many files available for downloading from bulletin boards are now sent in this compacted form.

The Archive software is public domain and can often be found amongst the downloadable files on bulletin boards, or alternatively, you may obtain it direct from BEEBUG. Please phone us for details.

call (at &FFDD) performs a number of functions depending on the value passed to the 6502's accumulator via A%. We will be setting A%=5 to read a file's catalogue information.

All OSFILE calls use a common *parameter block* for both passing and receiving data, and this requires 18 bytes. The filename must also be stored somewhere, and we will allocate a further 11 bytes for this purpose. In this OSFILE call, the first two bytes of the parameter block are set to the address of the filename. Opposite is a procedure to read a file's catalogue entry, together with a short demo program which shows how the relevant information can be extracted (and displayed).

Note that load and execute addresses allow for up to four bytes each, and that all values displayed are represented in hexadecimal.

```
  10 REM Program Read File Demo
 100 DIM block 18,name 11
 110 REPEAT
 120 INPUT'"Filename: " name$
 130 PROCread_catalogue(name$)
 140 PRINT"Load address: ";~block!2
 150 PRINT"Execute address: ";~block!6
 160 PRINT"File size: ";~block!10
 170 PRINT"Lock status: ";~block!14
 180 UNTIL FALSE
 190 END
 200 :
1200 DEF PROCread_catalogue(filename$)
1210 $name=filename$
1220 ?block=name MOD 256
1230 block?1=name DIV 256
1240 A%=5
1250 X%=block MOD 256
1260 Y%=block DIV 256
1270 CALL &FFDD
1280 ENDPROC
```

The three system calls I have used as examples are all described in the User Guide for the model B and B+. More information is contained in publications such as *The New Advanced User Guide* by Dickens & Holmes price £19.95 and The *Advanced Disk User Guide* by Pharo at £14.95. Master and Compact users should refer to the *Reference Manual* parts one and two at £14.95 each. Part one is the most relevant in this case.

Technical Assistant

BEEBUG needs a Technical Assistant to work in our magazine department to provide technical support for both BEEBUG and RISC User.

The successful candidate should have a good knowledge of programming in Basic, and preferably an understanding of Assembler. Experience of using these languages on a BBC micro is highly desirable. The work will involve dealing with technical queries relating to our magazines, evaluating contributions, writing, editing and testing programs for use internally or for publication, and generally providing technical expertise and support.

If you are interested in working for a friendly but go-ahead organisation in an interesting and challenging job, then we would like to hear from you as soon as possible.

To apply, write to The Personnel Manager at the address below, giving full details of yourself, your education and work experience (if any), and convince us you are the right person for the job.

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

HINTS and tips

Another round of hints and tips compiled by Lance Allison. It is nice to see so many different hints being submitted on all aspects of the BBC micro. The author of every hint published here will receive five pounds and, as usual, the author of the star hint will receive fifteen pounds.

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

DOS LINEFEEDS
by Richard Grant

Users of the 512 Co-processor will have found that DOS convention is to output a linefeed along with a carriage return. Under normal configuration this will produce double spacing on the printer. The obvious way of rectifying this is to change the DIP switch settings when using DOS. However, a much tidier method is to switch off Auto Linefeed from software. Unfortunately, as many members will be aware, the *FX6 command is not effective from within DOS.

Richard Grant suggests that the printer should be programmed so that a line-feed is half its normal height. This means that two line feeds will in fact only appear as one. The way to implement this is to use the Epson ESC,"A",6 control code sequence.

```
VDU2,1,27,1,65,1,6,3
```

Obviously this command cannot be entered from within DOS, so it should be used in a boot file (or Basic program) before the co-processor is selected.

SPACE IN BASIC IV
by James McDowel

Be aware that whilst entering Basic programs using AUTO, Basic IV will not place a space on the screen immediately following the line number, unlike the previous versions of Basic.

Another observation is that Basic IV will always strip trailing spaces, but only strips leading spaces (between line number and instruction) if LISTO is not zero.

*TITLE IN ADFS
by Adrian Bishop-Laggett

The old DFS *TITLE command would only accept spaces in the title if it was enclosed in quotes, whereas the same routine in the ADFS is remarkably forgiving and will allow you to enter the title in many different ways. Thus:

```
*TITLE "Utility disc"
*TITLE "Utility disc (omitting quote)
*TITLE Utility disc (no quotes)
```

will produce no error message and will each result in your disc being correctly titled. However, there is a strange bug. If you use the legal DFS form, putting the title in quotes but omitting the space between the command and the first quote;

```
*TITLE"Utility disc"
```

the ADFS puts the first word wholly in upper case!

TIDY ASSEMBLER
by David Spencer

Assembler listings obtained by using LIST or LISTO from Basic are usually not as clear as they might be. If you need to print out assembler listings why not, rather than enabling the printer and listing the Basic program, make sure that the OPT command is set to output assembly to screen, and enable the printer before assembling the program. The final listing will then appear in a more orderly manner, complete with the corresponding machine code.

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: Lance Allison
Production Assistant: Yolanda Turuelo
Membership secretary: Mandy Mileham
Editorial Consultant : Lee Calcraft
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.

BEEBUG Ltd (c) 1988 ISSN - 0263 - 7561

Printed by Head Office Design (0782) 717161

Magazine Disc/Cassette

NOVEMBER 1988 DISC/CASSETTE CONTENTS

RUN THE SOLAR SYSTEM - use this program to plot the orbits of the planets for any time in the past or future.

MOTORIST'S ROUTE PLANNER - avoid those unexpected hold-ups with this program that maps out the route to take.

LABEL PROCESSOR - use this label preparation utility to design, print and create a library of labels.

GRAPHIC DESIGN WITH ASTAAD (Part 2) - the second instalment of Astaad 3, providing more graphic functions and a host of extra facilities.

BASIC PROGRAM RESEQUENCER - make editing Basic programs easier with this utility to move or copy program lines.

TRIUMPHS AND TRIBULATIONS OF A TYRO - the two programs from the article that can be used to customise your Master computer.

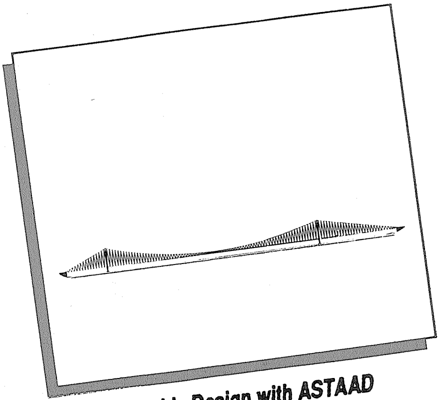
SOME NOVEL USES FOR THE MASTER MODEM - with a combination of the BEEBUG Master Modem and Superior Software's SPEECH! package, use your computer as a powerful answer phone.

BEEBUG WORKSHOP - this program to sort the words in a text file illustrates the use of a Binary Sequence Search Tree.

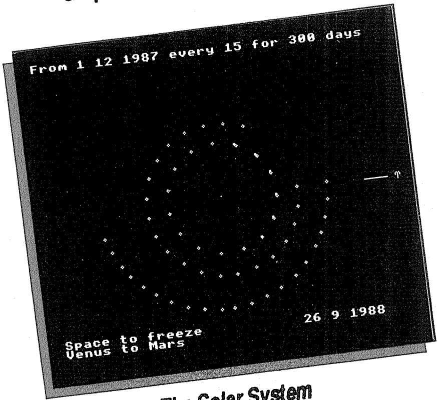
FIRST COURSE - a program to read the catalogue from a disc that shows how the CALL statement is used.

USING ASSEMBLER (Part 4) - an event driven clock that provides a continuous time display on the screen.

MAGSCAN - bibliography for this issue (Vol.7 No.6).



Graphic Design with ASTAAD



```
From 1 12 1987 every 15 for 300 days

                                26 9 1988
Space to freeze
Venus to Mars
```

The Solar System

All this for £3 (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.

	UK ONLY			OVERSEAS		
SUBSCRIPTION RATES	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:

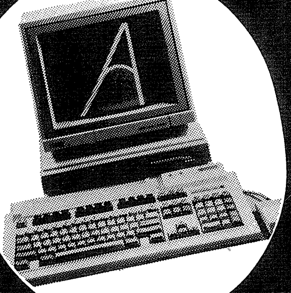
BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts. AL1 1EX.

Archimedes

Now Available from BEEBUG

featuring

THE NEW RISC OS



0% APR

Beebug the leading specialist dealers in Archimedes offer the best deals as well as comprehensive support for you and your computer.

We are Now able to offer–

0% Finance over 12 months

£50 Voucher to spend as you wish

FREE Printer Lead, Ten 3.5" discs, lockable disc box

and **FREE** RISC OS and Clares Artisan

or **FREE** PC Emulator

In addition, purchasers of a 440 system will receive a free copy of the First Word Plus Wordprocessor.

FREE RISC OS or PC EMULATOR

Subscribe to RISC USER (UK £14.50), the Archimedes magazine, and benefit from the additional FREE offers shown above. In addition you will have free access to our expert technical support team to assist and advise you.

0% FINANCE

Now over 12 months. Please phone for an application form. Return it with the deposit and we will be able to despatch your machine within 48 hours of receipt.

	Cost	Deposit		Cost	Deposit
305	801.60	**66.80**	310M	1027.00	**85.62**
305 Mono	870.60	**72.55**	310M Mono	1096.00	**91.37**
305 Colour	1054.60	**87.92**	310M Colour	1280.00	**106.74**
310	958.24	**79.87**	440	2906.10	**242.23**
310 Mono	1027.00	**85.62**	440 Colour	3159.10	**263.35**
310 Colour	1211.00	**100.88**			

Featuring Risc OS

The exciting new multi-tasking operating system for Archimedes, RISC OS, will be available early in 1989. Purchase an Archimedes under this scheme and you will be one of the first in the country to receive RISC OS, as soon as it becomes available. Your copy will be sent to you to install, or if you prefer, we will install it for you free of charge.

24 or 36 Months Credit and Trade ins

Should you prefer, we are also able to offer credit over 24 or 36 months at 13.75% flat rate (typical APR 27.5%). We are also able to offer up to £225 for your old BBC computer. Please ask for details.

Education and Health Discounts

Attractive discounts for schools, colleges and Health Authorities. Please ask for details.

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

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

Please indicate your requirements below.
Subscription to Risc User (£14.50 UK) ☐ Information Pack and Catalogue ☐ 0% Finance Form for 305/310/310M/440 Base/Mono/Colour ☐ 12/36 Months Finance Form for 305/310/310M/440 Base/Mono/Colour ☐ Trade In BBC/Master/Compact ☐ Purchase 305/310/440 Base/Mono/Colour ☐
UK Courier Delivery £7.00. Overseas please ask for a quotation. Prices include VAT

I enclose a cheque value £..............................

Please debit my Access/Visa/Connect Card No

Expiry....../...... with £............

Name ....

Address ....

....

Signature ....

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

BEEBUG – The Archimedes Specialists