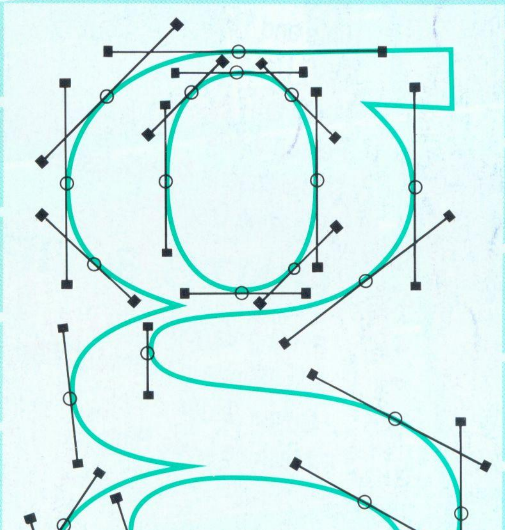
Vol.9 No.9 March 1991

BEEBUG

FOR THE
BBC MICRO &
MASTER SERIES

Beebug



Outline Font Designer

- BEEBUG FUNCTION/PROCEDURE LIBRARY
- CYCLIC FUNCTIONS
- CLUB MEMBERSHIP LISTS

BEEBUG Vol.9 No.9 March 1991

FEATURES

REVIEWS

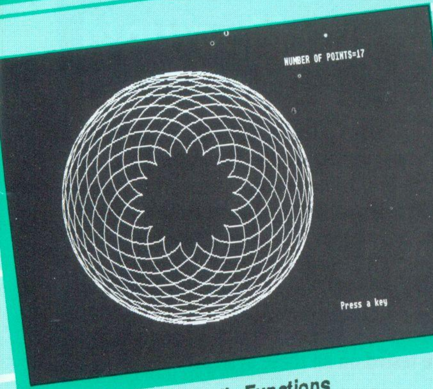
REGULAR ITEMS

HINTS & TIPS

PROGRAM INFORMATION

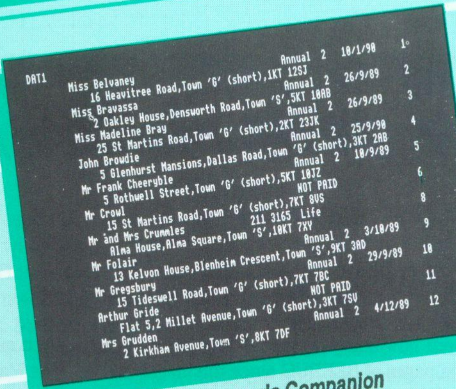
All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly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

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



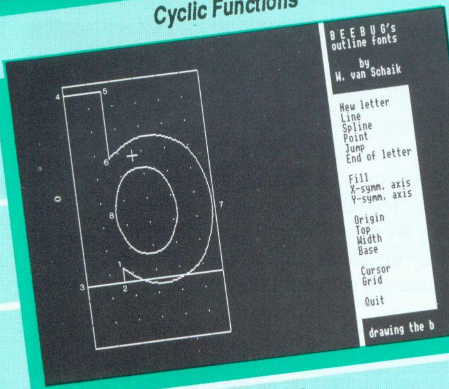
```
NUMBER OF POINTS=17

Press a key
```

Cyclic Functions

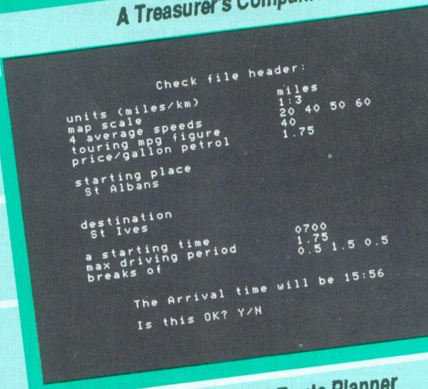


```
DAT1                                              Annual  2   10/1/90    1
     Miss Belvaney
        16 Heavitree Road,Town 'G' (short),1KT 12SJ
                                                  Annual  2   26/9/89    2
     Miss Bravassa
        2 Oakley House,Densworth Road,Town 'S',5KT 10AB
                                                  Annual  2   26/9/89    3
     Miss Madeline Bray
        25 St Martins Road,Town 'G' (short),2KT 23JK
                                                  Annual  2   25/9/90    4
     John Browdie
        5 Glenhurst Mansions,Dallas Road,Town 'G' (short),3KT 2AB
                                                  Annual  2   10/9/89    5
     Mr Frank Cheeryble
        5 Rothwell Street,Town 'G' (short),5KT 10JZ
                                                  NOT PAID               6
     Mr Crowl
        15 St Martins Road,Town 'G' (short),7KT 8VS
                                         211 3165  Life                  8
     Mr and Mrs Crummles
        Alma House,Alma Square,Town 'S',10KT 7XY
                                                  Annual  2   3/10/89    9
     Mr Folair
        13 Kelvon House,Blenheim Crescent,Town 'S',9KT 3AD
                                                  Annual  2   29/9/89   10
     Mr Gregsbury
        15 Tideswell Road,Town 'G' (short),7KT 7BC
                                                  NOT PAID              11
     Arthur Gride
        Flat 5,2 Millet Avenue,Town 'G' (short),3KT 7SV
                                                  Annual  2   4/12/89   12
     Mrs Grudden
        2 Kirkham Avenue,Town 'S',8KT 7DF
```

A Treasurer's Companion

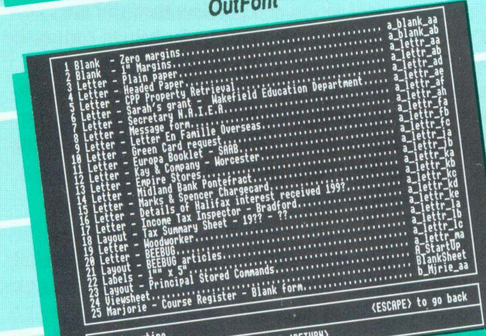


OutFont



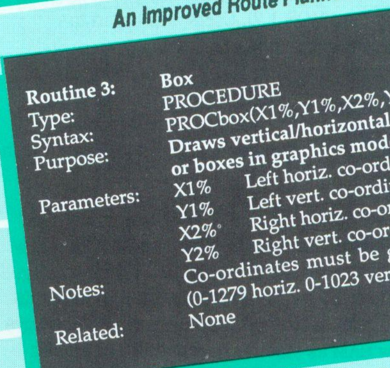
```
           Check file header:

units (miles/km)              miles
map scale                     1:3
4 average speeds              20 40 50 60
touring mpg figure            40
price/gallon petrol           1.75

starting place
  St Albans


destination
  St Ives
                              0700
a starting time               1.75
max driving period            0.5 1.5 0.5
breaks of

     The Arrival time will be 15:56

     Is this OK? Y/N
```

An Improved Route Planner



```
 1 Blank  - Zero margins.......................................... a_blank_aa
 2 Blank  - 1" Margins............................................ a_blank_ab
 3 Letter - Plain paper........................................... a_lettr_aa
 4 Letter - Headed Paper.......................................... a_lettr_ab
 5 Letter - CPP Property Retrieval................................ a_lettr_ad
 6 Letter - Sarah's grant -  Wakefield Education Department....... a_lettr_ae
 7 Letter - Secretary N.A.I.E.A................................... a_lettr_af
 8 Letter - Message form.......................................... a_lettr_ah
 9 Letter - Letter En Famille Overseas............................ a_lettr_fa
10 Letter - Green Card request.................................... a_lettr_fb
11 Letter - Europa Booklet - SAAB................................. a_lettr_fc
12 Letter - Kay & Company - Worcester............................. a_lettr_ja
13 Letter - Empire Stores......................................... a_lettr_jb
14 Letter - Midland Bank Pontefract............................... a_lettr_ka
15 Letter - Marks & Spencer Chargecard............................ a_lettr_kb
16 Letter - Details of Halifax interest received 199?............. a_lettr_kc
17 Letter - Income Tax Inspector - Bradford....................... a_lettr_kd
18 Letter - Tax Summary Sheet - 19?? - ??......................... a_lettr_ke
19 Layout - Woodworker............................................ a_lettr_la
20 Letter - BEEBUG................................................ a_lettr_lb
21 Letter - BEEBUG articles....................................... a_lettr_lc
22 Layout - 1"" x 5".............................................. a_lettr_ma
23 Labels - Principal Stored Commands............................. A_StartUp
24 Layout - Course Register - Blank form.......................... BlankSheet
25 Viewsheet...................................................... b_Mjrie_aa
   Marjorie
                                                  <ESCAPE> to go back

 Layout routine
 Key index (1 to 25) or "M" for more & <RETURN>
```

ADFS/View Utility for the Master



Routine 3:	**Box**
Type:	PROCEDURE
Syntax:	PROCbox(X1%,Y1%,X2%,Y2%)
Purpose:	**Draws vertical/horizontal lines or boxes in graphics modes**
Parameters:	X1% Left horiz. co-ordinate Y1% Left vert. co-ordinate X2% Right horiz. co-ordinate Y2% Right vert. co-ordinate
Notes:	Co-ordinates must be graphic (0-1279 horiz. 0-1023 vert.).
Related:	None

Function/Procedure Library

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

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

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

I hesitate to count how many programs have been published in the pages of BEEBUG since the magazine started back in 1983. This is the ninetieth issue of the magazine and I would guess that we must have published over 1000 programs during that period of time.

In the early days, matters were relatively simple. There was effectively only one machine, the original model B, and only one version of the operating system and of Basic (then Basic I). Programs in those early days were often much shorter in length than is now usually the case, so checking programs was a reasonably straightforward task.

Then, Acorn released Basic II, and a new version of the operating system (MOS 1.2). Users also began to enhance their systems, first of all with sideways ROM and RAM boards, and later with shadow RAM. Other suppliers produced alternative versions of the Beeb's filing system, often with double density formats for discs, and enhanced storage capabilities (compared with Acorn's DFS).

Eventually Acorn launched the Master 128, with a new version of the operating system (MOS 3.2) and a new version of Basic (Basic IV). This machine also supports the ADFS (as well as the DFS) and comes with sideways and shadow RAM as standard. Later still, the Master Compact appeared with further differences again between it and its predecessors. And in all of that, I have forgotten the short-lived model B+, a hybrid machine which was a stepping stone from the model B to the Master (but not upgradable as such).

With such a variety of hardware and software we have had to rationalise the systems on which we can test and support programs published in the magazine and on the magazine disc. For some long time now we have only been supporting Basic II and MOS 1.2 on the model B, The Master 128 and Master Compact, in all cases using Acorn supplied filing systems. As far as possible all programs are tested on all these systems.

We cannot cater specifically for proprietary sideways and shadow RAM boards which users might add to their model Bs, although programs which need these features may well work (sometimes with a few modifications) - we have to leave this to the judgement of the user. Neither can we take into account what other hardware or software individual users may have installed in their machines. We also have to work to a limited timescale, dictated by the publication schedule of the magazine, and occasionally something may be overlooked as a result.

We do provide support for all the programs which we publish, and if you have a problem then you are welcome to write or telephone us for help in sorting the problem out. Support of this kind is time consuming and expensive, so we would ask all readers to apply a little common sense and do all they can to determine the cause of the problem themselves before contacting us. It also helps to know and understand your own system and configuration as well as possible.

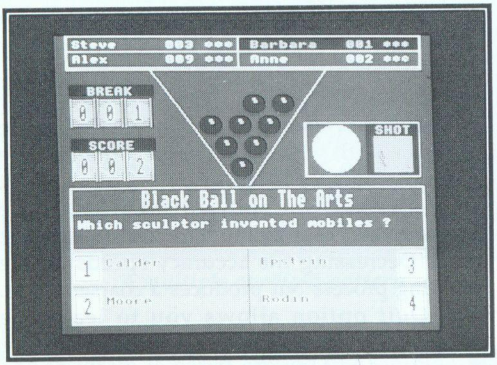
In particular we would urge all users to read carefully the accompanying information printed in the magazine for each program - it is surprising how often the answer to a question is there all the time. And do make use of the excellent manuals supplied for use with the various BBC systems - we cannot allow ourselves to be used as instantaneous talking manuals over the telephone; it is just too time consuming. Apart from that we will always do all we can to help you make the most of BEEBUG.

M.W.

News News News News News News

SUPERIOR GAMES SUPPORT

Games software house Superior continues to remain a staunch supporter of the BBC micro and its users with the release of a brand new game, *Master Break*. This is a snooker-style trivia game featuring six completely new and up-to-date sets of trivia questions in six categories: Science and Nature, Pop Music, Geography, Sports & Pastimes, Arts and History. *Master Break* can be played as a single player game, or with up to four players, with highest break, pass, and foul shot features.



With over 1500 questions (6000 answers) *Master Break* costs £9.95 for the cassette version, £11.95 for 5.25" disc and £14.95 for Master Compact 3.5" disc. A separate version, with more questions, is available for the Archimedes and A3000 for £19.95. All prices include VAT. The game is available through BEEBUG, or contact Superior Software, P.O.Box 6, Brigg, S.Humberside DN20 9NH, tel. (0652) 58585.

SHOWS FOR SHOPPERS

Ambitious Computer Shopper has announced two new shows for 1991, the *Spring Computer Shopper Show* will take place at Alexandra Palace from 10th to 12th May, and the *Christmas Computer Shopper Show* will again be at the Wembley Exhibition Centre from 5th to 8th December. According to information received from the show organisers, Acorn will have a large stand at both these events. A further computer show, the *European Computer Trade Show* will take place at the Business Design Centre, Islington, London from 14th to 16th April. All three shows are organised by Blenheim Database Exhibitions, Blenheim House, Ash Hill Drive, Pinner, Middlesex HA5 2AE, tel. 081-868 4466.

THE SOUND OF MUSIC

Another doyen of the BBC micro market, Cambridge-based Hybrid Technology, continues to develop new products in the music field. The latest product announced by this company is the *Music Publisher*, a powerful desktop music publishing system, which handles automatically the complex layout of music scores, ensuring high quality results without the time-consuming symbol positioning required by previous programs.

Also new in the SoundWorlds series (which includes *Soundshow* and *Soundscore* referred to in last month's News page) are *SoundStory*, a program aimed at primary schools which allows children to create written text and record music and sound effects, and combine them to produce musical stories, drama soundtracks, etc., and *Soundscene*, which links sound with interactive graphics to help pupils with learning difficulties and/or sensory handicaps explore a wide range of sounds using a touchscreen, joystick or switch.

Further new releases, shown at the BETT show at the Barbican in January, include the *Music 6000 Sensor*, a sonar-based unit which detects the position or movement of the body with great accuracy, allowing a computer to be controlled from distances up to 35 feet without any physical contact. Hybrid claims that the Music 6000 will be particularly valuable in linking music making with movement and dance, and for physically disabled pupils.

Units in the SoundWorlds series cost £29.00 each and the Music 6000 (which also needs the Music 5000 and software) costs £139.00 (all prices excluding VAT). For more information contact Hybrid Technology Ltd. at 273 The Science Park, Cambridge CB4 4WE, tel. (0223) 420360.

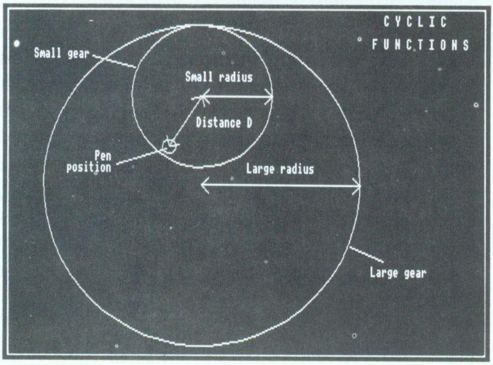
CASS FOR THE PRIMARY CURRICULUM

Curriculum Analysis Support System (CASS) is a new suite of programs produced by Doncaster based Resource which enable teachers to analyse their existing practice in terms of the National Curriculum Statements of Attainment. CASS uses a double-sided 80 track ADFS disc to run on a Master 128. A Classroom Pack costs £29.95 and a School Pack £95.00 (both prices exclusive of VAT). For more information contact Resource at Exeter Road, Doncaster DN2 4PY, tel. (0302) 328735.

Cyclic Functions

by Robert Templeman

This program is based on the popular graphics 'game' called Spirograph. The program will draw numerous cyclic shapes such as circles, ellipses, cardioids, deltoids and many more intricate shapes. Using the program is simple; all it requires are three numbers which can take integral values from 1 upwards.



The arrangement of the wheels of a Spirograph, on which this program is based

From the diagram it can be seen that the Spirograph consists of two gear wheels, one with teeth on the inside and one with teeth on the outside. The inner gear engages with the inside of the outer one and is turned by a pen inserted off-centre in the inner gear. This then traces a curved line which eventually meets up with itself and draws a cyclic function. The type of shape depends entirely on the ratio between the numbers of teeth of each gear, and the distance between the pen and the centre of the inner gear. The original Spirograph game can be quite tedious, and the pen is prone to slipping, but with this program it's easier, and quicker too.

Having typed the program in, save it with the name *Cyclic*, and type RUN. The program will draw several small cyclic functions and confront you with a menu.

Use Space to move through the menu, and Return to select your choice. The three options are *Design cyclic function*, *Alter accuracy of pattern* and *Quit*. The first of these, when selected, draws an explanatory diagram as shown above, and asks you to enter the three numbers. You will be asked for the radius of each gear; this is directly proportional to the number of teeth in the wheel. The values you enter will be used directly as screen co-ordinate values, so you can get an idea of how big the shape will be (e.g. if the large radius is 500 this will approximately fill the screen). When the shape is drawn, the computer tells you how many points there are in the shape.

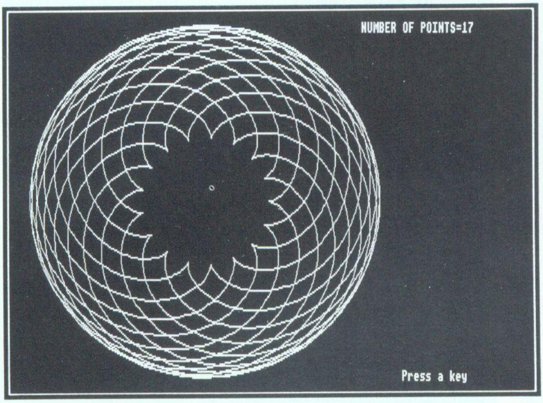
The *Alter accuracy* option allows you to alter the accuracy with which the shape is drawn. The computer draws a curve by drawing short straight lines that join up, so altering the accuracy will change the length of these short lines. Decreasing the accuracy speeds up the drawing process but produces a coarser shape. The *Quit* option allows you to leave the program without pressing Break. Pressing Escape anywhere in the program simply returns you to the menu.

EXAMPLES

Here are some values to try:

	Large radius	Small radius	Distance	Description
1	500	50	0	Circle
2	500	250	150	Ellipse
3	170	340	340	Cardioid
4	480	160	160	Deltoid
5	500	125	45	Square
6	500	100	40	Pentagon
7	500	300	200	5 points
8	250	400	400	5 points
9	500	285	250	10 points
10	205	360	360	41 points
11	560	250	180	56 points
12	500	285	250	100 points
13	400	202	300	200 points
14	400	201	300	400 points
15	5	500	500	1 point

Note that in examples 3,8 and 10, the small radius is larger than the large radius. This is not possible in the Spirograph game but is perfectly feasible in the program, and produces some interesting results. The last few examples take quite a while to draw and you might try altering the accuracy to a lower value to speed up the process.



```
NUMBER OF POINTS=17
Press a key
```

A multiple cardioid pattern

PROGRAM NOTES

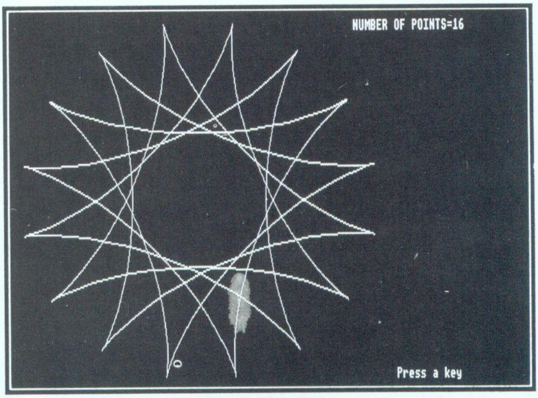
The program is based around one simple parametric formula (this appears at line 1180):

```
X=(b-s)*COS(t)-d*COS[(b/s-1)*t]
Y=(b-s)*SIN(t)+d*SIN[(b/s-1)*t]
```

where x and y are the x and y co-ordinates to plot, b is the radius of the large gear, d is the distance of the pen from the centre of the inner gear, s is the radius of the small gear and t is the parameter; varying the latter results in the appropriate x and y co-ordinates being calculated. The first terms in the formulae are the parametric forms of a circle of radius b-s. If d is set to zero the program then draws this circle. The second part of the formulae superimposes another circular function on the original; as the two are related by b and s this results in some very interesting shapes being drawn.

The rest of the maths is concerned with ensuring that the computer draws the entire shape. The simple cyclic functions such as circles require that t be varied from 0 to 2*PI radians (i.e. 0 to 360 degrees). However the more complex shapes require two or more rotations and the computer calculates this by reference to the ratio of b to s. If the ratio is an integer then only one full rotation is reqired (0 to 360 degrees) to draw the shape, but if the ratio is non-integer then the program multiplies it by integral values from 1 up to the largest gear radius (lines 1070 to 1100) until it gets an integer. It then uses the multiplier as the number of rotations required (multiples of 360 degrees). This ensures that the shape is drawn correctly.

One other interesting quantity is the number of points in the shape. This is the number of loops or points in the pattern and is simply the integer obtained from multiplying the b/s ratio. For instance, if the large gear has a radius of 500 and the small gear has a radius of 250 then the ratio of b/s is 2, and since this is an integer it requires only one rotation. The shape drawn has two points and is an ellipse. If d was zero then it would draw a circle.



```
NUMBER OF POINTS=16
Press a key
```

A 16 - point deltoid pattern

PROGRAM PROCEDURES AND VARIABLES

The important procedures are as follows:

PROCdraw(X%,Y%,lar%,sma%,D%,st%)
This procedure does all the work. X% and Y% are the co-ordinates of the centre of the shape,

lar% is the radius of the outer gear, sma% is the radius of the inner gear, D% is the distance of the pen from the centre of the inner gear, st% is the accuracy of the shape (10 is the default).

PROCcircle(X%,Y%,rad%)
draws a circle of radius rad% and centre co-ordinates X% and Y%.

PROCarrow(X1%,Y1%,X2%,Y2%)
draws an arrow from X1%,Y1% to X2%,Y2%.

PROCalter
allows you to alter the accuracy of the shape.

PROCsetup
sets up variables etc.

PROCbox
draws a border line on the screen.

VARIABLES

points%
the number of points in the shape, returned by PROCdraw.

max%
used by PROCdraw to store the largest radii when calculating the number of rotations required.

se & an
the step and the total angle by which the parameter t is varied.

I% & R%
constants set up to increase the speed of the program.

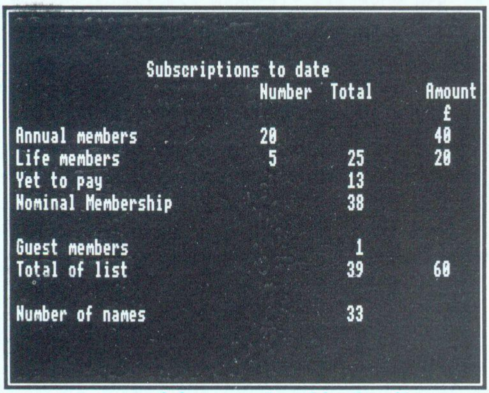
```
   10 REM          >Cyclic
   20 REM Program Cyclic Functions
   30 REM Version B1.0
   40 REM Author  Robert T. Templeman
   50 REM BEEBUG  March 1991
   60 REM Program subject to copyright
   70 :
   80 MODE0
   90 ONERROR IF ERR=17 VDU29;0;0:CLS:GO
TO 190 ELSE REPORT:PRINT" at line ";ERL:
END
  100 PROCsetup
  110 PRINTTAB(25,2);A$
  120 PRINTTAB(30,4)"By Robert Templeman
"
  130 PROCdraw(130,100,150,75,0,5)
  140 PROCdraw(330,100,100,50,30,10)
  150 PROCdraw(530,100,30,60,60,20)
  160 PROCdraw(730,100,99,66,50,5)
  170 PROCdraw(930,100,100,75,60,5)
  180 PROCdraw(1130,100,100,90,50,5)
  190 REPEAT
  200 index%=13:oldindex%=19
  210 PRINTTAB(19,13);"D E S I G N   ";L
EFT$(A$,29)TAB(17,15);"A L T E R  A C C
 U R A C Y  O F  P A T T E R N"
  220 PRINTTAB(33,17);"Q U I T"TAB(17,24
);"Space to move cursor     Return to se
lect item"
  230 REPEAT
  240 PROCbox
  250 PRINTTAB(9,oldindex%);SPC(3)TAB(70
,oldindex%);SPC(3)
  260 PRINTTAB(9,index%);">>>"TAB(70,ind
ex%);"<<<"
  270 oldindex%=index%
  280 A=GET
  290 IF A=32 THEN index%=index%+2:IF in
dex%=19 THEN index%=13
  300 UNTIL A=13
  310 IF index%=15 THEN PROCalter:GOTO20
0
  320 IF index%=17 THEN end%=TRUE:GOTO54
0
  330 CLS
  340 PROCbox
  350 PROCcircle(512,512,450)
  360 PROCcircle(512,762,200)
  370 PROCcircle(420,620,20)
  380 PROCarrow(512,512,962,512)
  390 PROCarrow(512,762,712,762)
  400 PROCarrow(420,620,512,762)
  410 PRINTTAB(29,6);"Small radius"TAB(4
0,14);"Large radius"TAB(31,10);"Distance
 D"TAB(65,5);"Enter"TAB(65,6)"Large radi
us"
  420 PRINTTAB(64,1);LEFT$(A$,11)TAB(62,
3);RIGHT$(A$,17)
```

```
  430 INPUTTAB(70,7);lrad%
  440 PRINTTAB(65,9);"Enter"TAB(65,10)"S
mall radius"
  450 INPUTTAB(70,11);srad%
  460 PRINTTAB(65,14);"Enter"TAB(65,15)"
Distance D"
  470 INPUTTAB(70,16);d%
  480 CLS
  490 PROCbox:IF srad%>lrad% THEN tempst
ep%=10*step%:ELSE tempstep%=step%
  500 PROCdraw(512,512,lrad%,srad%,d%,te
mpstep%)
  510 PRINTTAB(58,1);"NUMBER OF POINTS="
;points%TAB(65,30);"Press a key"
  520 *FX15,1
  530 a=GET
  540 CLS
  550 UNTIL end%=TRUE
  560 END
  570 :
  580 DEF PROCbox
  590 MOVE 0,0:DRAW 1279,0:DRAW 1279,102
3:DRAW 0,1023:DRAW 0,0
  600 ENDPROC
  610 :
  620 DEF PROCdraw(X%,Y%,lar%,sma%,D%,st
%)
  630 max%=lar%
  640 IF sma%>lar% THEN max%=sma%
  650 FOR N%=1 TO max%
  660 Q=N%*lar%/sma%
  670 IF Q=INT(Q) THEN points%=Q:R%=N%:N
%=max%
  680 NEXT N%
  690 PROCbox
  700 VDU29,X%;Y%;
  710 se=PI*R%/(st%*points%)
  720 an=2*PI*R%+se
  730 I%=lar%-sma%:R=lar%/sma%-1
  740 MOVE I%-D%,0
  750 FOR T=0 TO an STEP se
  760 DRAW I%*COS(T)-D%*COS(T*R),I%*SIN(
T)+D%*SIN(T*R)
  770 NEXT T
  780 VDU29,0;0;
  790 ENDPROC
  800 :
  810 DEF PROCcircle(X%,Y%,rad%)
  820 VDU29,X%;Y%;
  830 MOVE rad%,0
  840 se=2.2/SQR(rad%)
  850 FOR T=0 TO 6.4 STEP se
  860 DRAW rad%*COS(T),rad%*SIN(T):NEXT
  870 VDU29,0;0;
  880 ENDPROC
  890 :
  900 DEF PROCarrow(X1%,Y1%,X2%,Y2%)
  910 TH=ATN((Y1%-Y2%)/(X1%-X2%))
  920 MOVE X1%+30*COS(TH+PI/4),Y1%+30*SI
N(TH+PI/4)
  930 DRAW X1%,Y1%
  940 DRAW X1%+30*COS(TH-PI/4),Y1%+30*SI
N(TH-PI/4)
  950 MOVE X1%,Y1%:DRAW X2%,Y2%
  960 MOVE X2%-30*COS(TH-PI/4),Y2%-30*SI
N(TH-PI/4)
  970 DRAW X2%,Y2%
  980 DRAW X2%-30*COS(TH+PI/4),Y2%-30*SI
N(TH+PI/4)
  990 ENDPROC
 1000 :
 1010 DEF PROCalter
 1020 VDU 28,1,24,78,13:CLS:VDU26
 1030 PRINTTAB(10,13);"Increasing the ac
curacy of the shape will result"
 1040 PRINTTAB(10,14);"in a smoother cur
ve being drawn."
 1050 PRINTTAB(10,15);"This will, howeve
r, slow the program down."
 1060 PRINTTAB(10,17);"Default value is
10. Current value is ";step%"."
 1070 INPUTTAB(10,19);"Enter new step (R
eturn=default): "step%:IF step%=0 THEN s
tep%=10
 1080 step%=ABS(step%)
 1090 VDU 28,1,24,78,13:CLS:VDU26
 1100 ENDPROC
 1110 :
 1120 DEF PROCsetup
 1130 end%=FALSE
 1140 step%=10
 1150 PROCbox
 1160 VDU23;8202;0;0;0;
 1170 MOVE 0,800:DRAW 1279,800:MOVE 0,20
0:DRAW 1279,200
 1180 A$="C Y C L I C   F U N C T I O N
S"
 1190 ENDPROC
```

A Treasurer's Companion

Donald Ambrose explains how he fulfils the responsibilities of membership secretary for his local society using his BBC micro.

I recently became Treasurer and Membership Secretary of a local society of about 400 members, and the first call on my BBC micro was for it to print labels for the postal distribution of a newsletter. I started on this as a word processing operation, and tried out the multi-column program in BEEBUG Vol.7 No.1 before noticing that I could print three columns to match the stationery I have with Interword.



```
         Subscriptions to date
                     Number  Total    Amount
                                         £
Annual members         20                40
Life members            5      25        20
Yet to pay                     13
Nominal Membership             38

Guest members                   1
Total of list                  39        60

Number of names                33
```

Summarising membership details

However, I also wanted to use my computer to keep the subscription record so I needed something more than word processing. I had not been favourably impressed by what my predecessor offered me from his computer as I thought its format was poor and mistakes had escaped notice because they were buried in a file where they were not visible. I was not sure what the software I could buy would do, but anyway, why pay money when I could write a program that was certain to give me what I wanted? Here is the result.

In distant times other people organized computing for me, and when micro-computers first arrived I had "user-friendly" programs. These delayed my understanding of what was going on for several years until I was on my own and had to organize the computer for myself. It was a considerable task to eliminate all the user-friendly features in order to identify the essential parts of the programs so that I could convert them for use on another computer. That experience suggested that while some programs may be so complex that no non-professional will understand them, or should even try to do so, it is better for a relatively simple program such as the present one to be kept simple so that it is easily comprehensible, and can be easily altered to suit the reader's own purposes.

SIMPLE MEMBERSHIP LISTS

I see no point in fancy menus; what comes here is as simple as possible. The program starts by asking if paged or non-paged mode is preferred and whether the printer is on or off (in effect whether or not printed output is also required). The main dialogue asks whether the user wishes to:

1. Print labels (70 mm by 37 mm) in three columns;

2. List the members with or without their addresses and count them;

3. List the members who have not yet paid.

A "count only" option is also provided so that the program can be run on the day of a committee meeting for an up-to-date report on the state of the membership. These permutations of the possibilities are what suit me and it will not be difficult for the reader to alter them if some others are preferred. I have a monochrome monitor and always use mode 131 (this is on a Master); line 2070 of the program reveals also that I operate in paged mode.

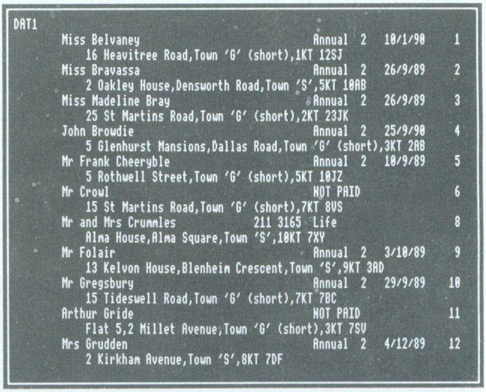
Another aspect of my practice is that I avoid having anything to do with "files", meaning by "files" the burying of the data in bytes that can only be unravelled by special file-reading programs. I prefer to have all my data in DATA lines so that everything is in memory, the DATA can be READ by the standard Basic command, and the DATA can be edited as part of the program (or as text in a word processor or in my case using the Master's Edit). It seems to me that the only objection to doing this is that memory may be insufficient but, as this program shows, that problem can also be overcome (and will help in running the program on a model B as well).

In fact, I find that there is room for somewhat over 100 entries in addition to the program if Edit is to be usable, and a membership of 400 (some are married couples so it is not a question of simple arithmetic) requires that I load my data in three blocks. The three blocks are appended successively and automatically to the program, and I have made up three small blocks of names and addresses, DAT1, DAT2, DAT3, to demonstrate how it works (DAT1 is listed here for reference; the other two are also included on the magazine disc).

The names are entered in alphabetical order, and there is no difficulty in adding new names in their correct position (and renumbering the lines as required). Loading the blocks of DATA is achieved by lines 130 to 190 of the main program, and lines 280 to 300 clear the program of all DATA by loading *Blank*, which is simply the name of a program containing nothing. I would never answer "Yes" to the question "PRINTER ON?" without having checked on screen that all is well, so it is useful to be able to restore everything to its initial state for re-running, and PROCclear deals with this.

ENTERING THE MAIN PROGRAM

Type in the program given in listing 1 and save as *Memlis*. Most articles in this magazine have a ritual injunction to save the program before running; I recommend doing so long before getting to that point, and myself save every quarter of an hour or so, whatever I am typing. That may seem excessive, but I learnt to do it in the days when networks broke down frequently and an hour's work could easily be lost through failing to save. If you do not SAVE there is also the ghastly possibility that RUN will produce "Bad Program" - and then all is lost unless you are kitted up with a *Bad Program Lister*.



```
DAT1
        Miss Belvaney                                   Annual  2   10/1/90     1
            16 Heavitree Road,Town 'G' (short),1KT 12SJ
        Miss Bravassa                                   Annual  2   26/9/89     2
            2 Oakley House,Densworth Road,Town 'S',5KT 10AB
        Miss Madeline Bray                              Annual  2   26/9/89     3
            25 St Martins Road,Town 'G' (short),2KT 23JK
        John Browdie                                    Annual  2   25/9/90     4
            5 Glenhurst Mansions,Dallas Road,Town 'G' (short),3KT 2AB
        Mr Frank Cheeryble                              Annual  2   10/9/89     5
            5 Rothwell Street,Town 'G' (short),5KT 10JZ
        Mr Crowl                                        NOT PAID                6
            15 St Martins Road,Town 'G' (short),7KT 8VS
        Mr and Mrs Crummles                    211 3165 Life                    8
            Alma House,Alma Square,Town 'S',10KT 7XY
        Mr Folair                                       Annual  2   3/10/89     9
            13 Kelvon House,Blenheim Crescent,Town 'S',9KT 3AD
        Mr Gregsbury                                    Annual  2   29/9/89    10
            15 Tideswell Road,Town 'G' (short),7KT 7BC
        Arthur Gride                                    NOT PAID               11
            Flat 5,2 Millet Avenue,Town 'G' (short),3KT 7SV
        Mrs Grudden                                     Annual  2   4/12/89    12
            2 Kirkham Avenue,Town 'S',8KT 7DF
```

Listing members and addresses

CREATING THE DATA FILES

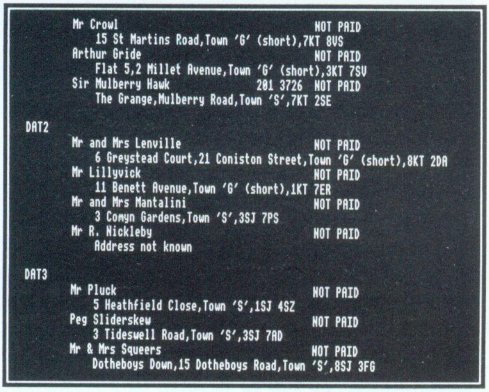
Each block of data statements should be entered and saved as for a Basic program - see sample file given in Listing 2 called DAT1. To work with the main program as listed you will need to spread your data over three such files called DAT1, DAT2 and DAT3, each numbered so that they can be appended to Memlis successively (e.g. starting at lines 3000, 4000 and 5000).

The DATA lines for each member contain 10 string variables, and I have laid them out with two on one line (number of members for the entry, name) and 8 on the next (four for address including the postcode, telephone number, date paid, amount, and whether Annual or Life). So we have:

```
X000 DATA number,name
Y000 DATA ad1,ad2,ad3,ad4,phone,date
,sub,L or A
```

where *ad1, ad2, ad3, ad4* are the components of the address, and 'L' or 'A' indicates Life or Annual member. We have several "guests" who do not pay subscriptions, and these are dealt with by making "num$ (number) for them negative. Each block ends with a DATA line with the one value of 100, which is READ as num$ and triggers loading the next block.

For economy in memory, and simplicity in typing, the two towns where our members live are abbreviated in the DATA lines to 'G' and 'S'. One of these ('G') has two expanded forms - one complete on the labels for the Post Office and, since we know where we all live, one incomplete for the membership list. The full names are set in lines 240 and 250 in variables C1$ (town 'G' long form), C2$ (town 'S' long form), and C3$ (town 'G' short form). If any of the ten variables is null that must be indicated by consecutive commas.



```
        Mr Crowl                              NOT PAID
           15 St Martins Road,Town 'G' (short),7KT 8VS
        Arthur Gride                          NOT PAID
           Flat 5,2 Millet Avenue,Town 'G' (short),3KT 7SV
        Sir Mulberry Hawk           201 3726  NOT PAID
           The Grange,Mulberry Road,Town 'S',7KT 2SE

DAT2
        Mr and Mrs Lenville                   NOT PAID
           6 Greystead Court,21 Coniston Street,Town 'G' (short),8KT 2DA
        Mr Lillyvick                          NOT PAID
           11 Benett Avenue,Town 'G' (short),1KT 7ER
        Mr and Mrs Mantalini                  NOT PAID
           3 Comyn Gardens,Town 'S',3SJ 7PS
        Mr R. Nickleby                        NOT PAID
           Address not known

DAT3
        Mr Pluck                              NOT PAID
           5 Heathfield Close,Town 'S',1SJ 4SZ
        Peg Sliderskew                        NOT PAID
           3 Tideswell Road,Town 'S',3SJ 7AD
        Mr & Mrs Squeers                      NOT PAID
           Dotheboys Down,15 Dotheboys Road,Town 'S',8SJ 3FG
```

List of members not yet paid

It is difficult to type in details of 400 members without mistakes, and I first tried checking my layout by using COMMAS from BEEBUG Vol.7 No.1, but then decided that three lines of program at the beginning of each data block would do the job better. These lines have no effect on the main program but allow each block to be run (and I do every time I make an amendment), and if it delivers a column of names on the screen it is likely that all is in order.

Mistakes most easily arise because of an incorrect number of commas in the second line, and the best way of avoiding this is to set AUTO at the appropriate number and hold down a key programmed as:

```
*KEY9 DATA ,|MDATA ,,,,,,,|M.
```

The result will be a series of numbered DATA lines, and when sufficient have appeared the details can be filled in between the commas.

Handling data in this way has one disadvantage and one advantage. The disadvantage is that since each line is numbered, carelessness in numbering when amendments are being entered can lead to a line being overwritten, and the possibility of this happening and not being noticed is reduced by using two lines for each entry rather than only one. The advantage is that REM statements allow unformatted annotations to be included among the data.

Come the end of the financial year and the Treasurer needs to start all over again. If Edit (or similar) is available, the DATA blocks can be spooled to text files and then edited as text. The parts of the DATA statements ",x," and ",y," (where x and y are the subscriptions) need to be globally changed to ",," or ",0,". As subscriptions are paid each member's record can be updated and amended accordingly. It will be noted that my subscriptions are all whole pounds and treated as integers. If any pennies are involved a few changes will need to be made (or multiply/divide by 100 internally so that integers - i.e. amounts in pence - can still be used).

The file *Blank* referred to previously can be created by typing:

```
NEW
SAVE "Blank"
```

Finally, to conform with the law - Section 33 (2) of the Data Protection Act 1984 - you must inform your members that they are to become "data subjects", and persuade them not to object to being listed on your computer.

Editorial Note: We feel that the experiences described here may well be of interest to other readers, but do remember that the program has been written to suit particular circumstances. The intention is that readers should adapt this program to their own needs.

Listing 1

```
   10 REM Program Memlis
   20 REM Version B1.0
   30 REM Author  Donald Ambrose
   40 REM BEEBUG  March 1991
   50 REM Program subject to copyright
   60 :
  100 MODE 3:B%=0
  110 PROCclear:C%=TOP-2
  120 PROCchoice
  130 IF B%=0 f$="DAT1"
  140 IF B%=1 f$="DAT2"
  150 IF B%=2 f$="DAT3"
  160 VDU3:PRINT'f$:VDU21
  170 *KEY0 OS.("LOAD "+f$+" "+STR$~C%)|
MOLD|MGOTO210|M
  180 *FX138 0 128
  190 END
  200
  210 DIM name$(3),ad$(12)
  220 B%=B%+1:VDU6
  230 ON ERROR GOTO 390
  240 C1$="Town 'G' (long)"
  250 C2$="Town 'S'":C3$="Town 'G' (shor
t)"
  260 PROCrun
  270 IF B%<3 GOTO 130
  280 f$="BLANK":VDU3
  290 *KEY0 OS.("LOAD "+f$+" "+STR$~C%)|
MOLD|M:VDU6|M
  300 *FX138 0 128
  310 IF J%=5 GOTO 360
  320 IF J%=1 VDU3:PRINT "Number of addr
essees "K%
  330 IF J%=1 PRINT "Number of labels
   "M%
  340 IF H% VDU2
  350 IF J%>1 PROCtot
  360 PROCclear:VDU21
  370 END
  380 :
  390 PROCclear
  400 REPORT:PRINT" at line ";ERL
  410 END
  420 :
 1000 DEF PROCrun
 1010 IF H% VDU2 ELSE VDU3
 1020 IF J%=1 PROClabel
 1030 IF J%>1 PROClist
 1040 VDU3:ENDPROC
```

```
 1050
 1060 DEF PROCchoice
 1070 VDU3:PRINT "PAGE MODE ON?"
 1080 A$=GET$:IF A$="Y" VDU14 ELSE VDU15
 1090 PRINT "PRINTER ON?": H%=(GET$="Y")
 1100 PRINT'"LABELS";SPC25;"(1) ?"
 1110 PRINT "COUNT";SPC26;"(2) ?"
 1120 PRINT "LIST MEMBERS WITH ADDRESSES
    (3) ?"
 1130 PRINT "LIST MEMBERS WITHOUT ADDRES
SES (4) ?"
 1140 PRINT "LIST MEMBERS NOT PAID ONLY
    (5) ?"
 1150 PRINT'SPC10;"TYPE NUMBER OF CHOICE
"
 1160 J%=GET-48
 1170 ENDPROC
 1180
 1190 DEF PROClist
 1200 REPEAT PROCreadlist
 1210 IF J%<>2 PROCprintlist
 1220 UNTIL A%=1
 1230 IF J%=5 GOTO 1250
 1240 VDU3:IF B%=3 PRINT'"Press any key
for summary":A=GET:CLS
 1250 ENDPROC
 1260
 1270 DEF PROClabel
 1280 REPEAT PROCreadprint:UNTIL A%=1
 1290 ENDPROC
 1300
 1310 DEF PROCtot
 1320 PRINT
 1330 PRINTTAB(30)"Subscriptions to date
"
 1340 PRINTTAB(43)"Number"TAB(51)"Total"
TAB(62)"Amount"
 1350 PRINTTAB(64)"£"
 1360 PRINTTAB(15)"Annual members"TAB(35
)P%TAB(55)T%
 1370 PRINTTAB(15)"Life members"TAB(35)L
%TAB(45)P%+L%TAB(55)V%
 1380 PRINTTAB(15)"Yet to pay"TAB(45)K%-
P%-L%-F%
 1390 PRINTTAB(15)"Nominal Membership",T
AB(45)K%-F%:PRINT
 1400 PRINTTAB(15)"Guest members",TAB(45
)F%
 1410 PRINTTAB(15)"Total of list"TAB(45)
K%TAB(55)T%+V%:PRINT
 1420 PRINTTAB(15)"Number of names"TAB(4
5)M%
 1430 PRINT:ENDPROC
 1440
 1450 DEF PROCreadlist
 1460 A%=0:READ num%
 1470 IF num%>=100 num%=0:A%=1:ENDPROC
 1480 IF num%<0 num%=-num%:F%=F%+num%
 1490 K%=K%+num%:M%=M%+1
 1500 READ name$,ad$(1),ad$(2),ad$(3),ad
$(4)
 1510 READ phone$,date$,sub$,AorL$:S%=VA
L(sub$)
 1520 IF AorL$="L" AorL$="Life"
 1530 IF AorL$="Life" AND sub$="0" sub$=
""
 1540 IF AorL$="Life":V%=V%+S%:L%=L%+num
%
 1550 IF AorL$="A" AorL$="Annual"
 1560 IF AorL$="Annual" AND sub$="0" sub
$=""
 1570 IF AorL$="Annual" AND sub$="" AorL
$="NOT PAID"
 1580 IF AorL$="Annual" T%=T%+S%:IF S%>0
 P%=P%+num%
 1590 IF AorL$="F" AorL$="Free":sub$=""
 1600 ENDPROC
 1610
 1620 DEF PROCprintlist
 1630 IF A%=1 ENDPROC
 1640 IF J%=5 AND AorL$<>"NOT PAID" ENDP
ROC
 1650 IF ad$(2)="G"  ad$(2)=C3$ ELSE IF
ad$(3)="G"  ad$(3)=C3$
 1660 IF ad$(2)="S"  ad$(2)=C2$ ELSE IF
ad$(3)="S"  ad$(3)=C2$
 1670 @%=&15:PRINTTAB(8)name$;
 1680 @%=&0A:PRINTTAB(40)phone$TAB(50)Ao
rL$;:@%=&03:PRINTTAB(58)sub$;
 1690 @%=&0A:PRINTTAB(62)date$;:IF J%=5
GOTO 1710
 1700 @%=&03:PRINTTAB(72)K%
 1710 IF J%=4 GOTO1770
 1720 IF ad$(1)="" PRINTTAB(12)"Address
not known":GOTO1770
 1730 IF ad$(2)="" PRINTTAB(12)ad$(1):GO
TO1770
 1740 IF ad$(3)="" PRINTTAB(12)ad$(1)","
ad$(2):GOTO1770
 1750 IF ad$(4)="" PRINTTAB(12)ad$(1)","
ad$(2)","ad$(3):GOTO1770
 1760 PRINTTAB(12)ad$(1)","ad$(2)","ad$(
3)","ad$(4)
 1770 ENDPROC
 1780
 1790 DEF PROCreadprint
 1800 FOR I%=1 TO 12: ad$(I%)="":NEXT
 1810 FOR I%=1 TO 3:name$(I%)="":NEXT
 1820 A%=0:FOR I%=1 TO 9 STEP 4
 1830 READ num%:IF num%>=100 num%=0:A%=1
:IF I%=1 A%=2
 1840 IF A%>0 I%=10:GOTO 1900
 1850 IF num%<0 num%=-num%
 1860 K%=K%+num%:M%=M%+1
 1870 READ name$(I%/4+3/4),ad$(I%)
 1880 READ ad$(I%+1),ad$(I%+2),ad$(I%+3)
 1890 READ phone$,date$,sub$,AorL$
 1900 NEXT I%:IF A%=2 ENDPROC
 1910 FOR I%=2 TO 10 STEP 4
 1920 IF ad$(I%)="G" ad$(I%)=C1$
 1930 IF ad$(I%)="S" ad$(I%)=C2$
 1940 IF ad$(I%+1)="G" ad$(I%+1)=C1$
 1950 IF ad$(I%+1)="S" ad$(I%+1)=C2$
 1960 NEXT
 1970 PRINT:PRINT
 1980 PRINTTAB(2)name$(1) TAB(32)name$(2
) TAB(62)name$(3)
 1990 FOR I%=1 TO 4
 2000 PRINTTAB(2)ad$(I%) TAB(32)ad$(I%+4
) TAB(62)ad$(I%+8)
 2010 NEXT
 2020 ENDPROC
 2030
 2040 DEF PROCclear
 2050 B%=0:E%=0:F%=0:H%=0:J%=0:K%=0
 2060 L%=0:M%=0:P%=0:S%=0:T%=0:V%=0
 2070 VDU3,14
 2080 ENDPROC
 2090 :
 2100 REM B%=marker for data blocks
 2110 REM E%=line counter
 2120 REM H%=print marker
 2130 REM J%=number in PROCchoice
 2140 REM K%=total of list
 2150 REM L%=number of life
 2160 REM members, M%=no. of labels
 2170 REM num%=no. of each entry,
 2180 REM negative for guest members
 2190 REM P%=number of paid annual membe
rs
 2200 REM S%,T%(annual),V%(life)=subscri
ptions collected
```

continued on page 46

OutFont: An Outline Font Designer

Our previous program for printing spline (or outline) text has proved very popular. Now Willem van Schaik has produced a program which enables you to design your own outline fonts for use with that program, all on a BBC micro.

INTRODUCTION

BEEBUG for June 1990 (Vol.9 No.2, updated Vol.9 No.4) presented the character printing program Spline Text using outline fonts. With this program letters can be scaled and stretched before they are printed. The character set given with the program consisted of only the upper case characters plus comma, point and exclamation mark.

The need to extend this program with the lower case characters is a point which has already been addressed in BEEBUG. However, there is not only the matter of extending the character set, but the question of using different fonts is an issue that also comes up frequently, particularly with the increasing popularity of DTP. I therefore changed my original plan to extend the character set with lower-case characters into the development of a program to make the design of new fonts very easy.

OUTLINE FONT DESIGN

The design of letters using outline fonts is most often done with a digitiser. After the letter has been sketched on paper, the co-ordinates of the contours are entered into the computer just by pointing to them. However, few BEEBUG readers are likely to possess such a device.

Fortunately there is a "poor man's digitiser". The only thing needed is the possibility of using a photo-copying machine which can make copies on transparent sheets. The trick consists of making a photocopy of the drawing to be digitized on a sheet, then fitting this transparent sheet to the screen of the computer monitor, after which any CAD software package can be used to trace the drawing. This method can be used, for example, to digitize maps, but it can be used for fonts as well. In fact, if the initial drawing paper is sufficiently thin (and transparent) then this might suffice in itself.

When fonts are to be digitized, a normal CAD package is of limited use. The co-ordinates are always needed in a different format from that used by the CAD software. Hence a special program which helps the process of tracing the character outlines, and which delivers the co-ordinates in the format used by the program that uses the fonts is much better. *OutFont* listed here is such a program, and it creates a text file with DATA statements which can be appended to the *Spline2* program (from Vol.9 No.2) just by EXECing it.

HOW TO DO IT

Letter types can be very easily obtained from sheets of rub-down transfer letters. After choosing the right font, you can use the enlarging possibilities of a photocopying machine to get originals with a font height a little smaller than the monitor screen height. Then, maybe after using scissors and glue, make photocopies on transparency sheets (acetate sheet).

Now run the program (having typed it in and saved it) and you will see a square drawing area and a menu of the possible functions. Choose any function by typing the first character of the corresponding command. You can move the cursor with the standard cursor keys, movement being speeded up by holding down the Shift key at the same time.

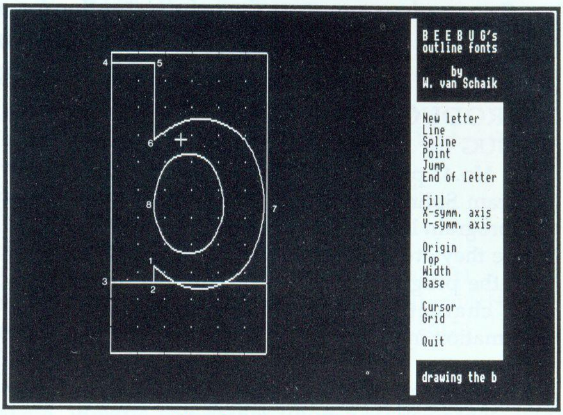
A number of help lines can be displayed on the screen with first, of course, the rectangle that encloses the letter. The lower-left corner of this rectangle is called the *Origin* and the upper side Top. The lower side of this rectangle is not the bottom of an "A" or "a", but the bottom point of a letter with descenders like "p" or "g". In addition, there is a so-called *Base* line, and this marks the bottom of a letter without descenders like "A" or "a". For all the characters in one font, the placement of the Base relative to the Origin and Top should remain constant. The width of the character - the right-hand edge of the enclosing rectangle - can and should be changed for each character.

EXAMPLE

Explaining all the functions is best done using an example. Let's suppose you are going to digitize the letter "b". First mark on the sheet where the origin of a letter "p" would have been, and then fit the sheet to the monitor using tape. Now press 'C' and the cursor will change to a large hair-line cursor. Align this with the left-hand side of the "b" and the low side of the "p" and press 'O' (for origin). Go to the top of the "b" and press 'T' (for top).

Pressing 'N' indicates the start of a new character, so answer the question with 'b'. Go to point 7 (see figure 1) and press 'W' (to specify the character's width). Now it's time to press 'C' again for a normal cursor and Shift-G for a grid. Pressing 'G' without the Shift key would produce a very fine grid. Before we start defining the outline we enter the fill-point by pressing the 'F' (the point from which the outline will be filled to produce a solid black character.

Move the cursor to 1 and press 'L' to start the polygon. Go to 2, 3, 4 and 5 each time pressing 'P' to enter a point, and you will see a straight line being drawn to link one point to the next. Now lower the cursor to 6 and press 'S' to start a spline curve. Choose a number of points on the curve each time pressing 'P'. At this stage the points will be linked with straight lines giving an approximate outline for the subsequent curve. Try to use as few points as possible. When you return to point 1 press 'J' to jump to the inner ellipse. Move the cursor to point 8 on the curve and press 'S' for a new spline. Enter a number of points and return to 8. To finish the character definition press 'E' and the co-ordinates will be put into the DATA format used by the program Spline2 and appended to the file *Font*. At the same time the resultant character will be redrawn, filled and displayed on the screen.



Tracing out a letter 'b'

You can now continue with the next letter by pressing 'N' again, or leave the program by pressing 'Q'. In this example all possible functions were used except for 'X' and 'Y'. With these functions two axes of symmetry can be defined, and an outline will be mirrored about the selected axis of symmetry. However, symmetry is generally only useful for simple fonts.

THE DATA LINES

In the original Spline2 program, the sequence of characters in the string in line 1350 combined with a corresponding order of the DATA lines (2620 and up) caused the selection of the right data for a certain character. The DATA lines produced by OutFont use a different method for identifying the right data.

The data for each character can be found by adding 10000 to the ASCII value of the character, which gives the line number of the corresponding DATA statement. This method has the advantage that adding or deleting character definitions can be done very easily, without the need to change the program. Further, when memory limitations become a problem, only the DATA lines with the character definitions required at the time need be added to the Spline2 program.

The consequence is that the original Spline2 program must be altered. Line 1350 must be replaced by the line:

```
1350 RESTORE (10000 + ASC(letter$))
```

Further, the original lines 2620-2870 must be renumbered to the range 10065-10090 and the last three lines of that program must be renumbered as lines 10046, 10033 and 10044.

TYPING IN THE PROGRAM

You can avoid a lot of typing if you already have a copy of the program Spline2, because the procedures and functions used for drawing the lines and splines are used again in OutFont. To do this execute the following commands:

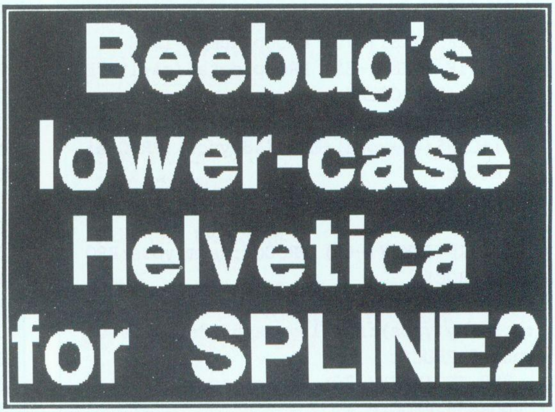
```
LOAD "SPLINE2"
DELETE 10,1450
DELETE 2270,2530
DELETE 2620,2900
RENUMBER 2350
SAVE "OUTFONT"
```

Now you have most of the last part of OutFont. In the lines 3060 to 3090 and in line 3210 you must make some small changes, and of course the rest of the new program (lines 10 to 2340) must also be typed in. Without Spline2 you will need to type in the complete listing.

In the program PAGE is lowered to &1200 (if higher than this initially), but it can still be useful to free more memory by compacting the program. The *SHRINK command in EdiKit will do this job very well. Take care not to save the compacted version using the same name as the original.

HELVETICA-BOLD

Using OutFont I digitized the full Helvetica-Bold character set with both upper and lower case characters, and the numerics. Because of memory limitations on un-extended BBC machines, you must make a selection of the DATA lines to be used. This can be best done with a word processor using a copy of Font. After deletion of the unnecessary statements save the file and load Spline2. Now delete the existing DATA lines and *EXEC the file just created with the DATA statements for the Helvetica-Bold font.



Beebug's
lower-case
Helvetica
for SPLINE2

Text display using Helvetica Bold created with OutFont

The magazine disc, in addition to the program OutFont, contains a copy of the previously published program Spline2 referred to in the text (modified as described), together with Spline1 (also from Vol.9 No.2), and an ASCII file Font containing the outline font definitions for the Helvetica-like character set as a set of DATA statements ready for appending to Spline2.

```
   10 REM OUTFONT Outline font designer
   20 REM Version B1.1
   30 REM Author  Willem van Schaik
   40 REM BEEBUG  March 1991
   50 REM Program subject to copyright
   60 :
  100 IF PAGE>&1200 THEN PAGE=&1200:CHAI
N"OutFont"
  110 ON ERROR OSCLI("FX4,0"):CLOSE#0:MO
DE 3:REPORT:PRINT" at line ";ERL:END
  120 DIM data$(50),X%(50),Y%(20),apeX%(
50),apeY%(50)
  130 MODE 0
  140 PROCinit:PROCopen:PROCfc
  150 REPEAT
  160 *FX21,0
  170 IF INKEY(-86) PROCcf:PROCnew:PROCf
c
```

```
  180 IF data$(0)="" GOTO 270
  190 IF INKEY(-87) PROCp:data$(s%+1)="L":s%=s%+2:PROCp
  200 IF INKEY(-82) PROCp:data$(s%+1)="S":s%=s%+2:PROCp
  210 IF INKEY(-56) PROCp
  220 IF INKEY(-70) PROCp:jump%=TRUE
  230 IF INKEY(-35) PROCp:data$(s%+1)="E":s%=s%+1:PROCend
  240 IF INKEY(-68) PROCcf:data$(2)=data$(2)+CHR$(48+xpos)+CHR$(48+ypos):PROCwait:PROCfc
  250 IF INKEY(-67) PROCcf:xsym=xpos:PROCsym:PROCfc
  260 IF INKEY(-69) PROCcf:ysym=ypos:PROCsym:PROCfc
  270 IF INKEY(-55) PROCcf:xor=FNx(xpos):yor=FNy(ypos):sc=(1-ypos/55)*sc:xpos=0:ypos=0:PROCfc
  280 IF INKEY(-36) PROCcf:sc=ypos/55*sc:ypos=55:PROCfc
  290 IF INKEY(-34) PROCcf:xwid=-1*xpos*(xpos<=55)-55*(xpos>55):PROCsym:PROCfc
  300 IF INKEY(-101) PROCcf:yba=ypos:PROCfc
  310 IF INKEY(-83) PROCc:hair%=NOT hair%:PROCc
  320 IF INKEY(-84) PROCgrid
  330 IF INKEY(-58) PROCmove(0,1)
  340 IF INKEY(-42) PROCmove(0,-1)
  350 IF INKEY(-26) PROCmove(-1,0)
  360 IF INKEY(-122) PROCmove(1,0)
  370 UNTIL INKEY(-17)
  380 PROCclose
  390 MODE 3
  400 *FX4,0
  410 *FX21,0
  420 END
  430 :
 1000 DEF PROCinit
 1010 *FX4,1
 1020 VDU 23,1,0;0;0;0;
 1030 VDU 28,66,31,79,8:PROCmenu
 1040 GCOL 0,0:GCOL 0,129
 1050 VDU 24,1040;799;1279;1023;:CLG
 1060 VDU 24,1040;0;1279;95;:CLG
 1070 VDU 24,0;0;1023;1023;:CLG
 1080 COLOUR 0:COLOUR 129
 1090 VDU 28,66,6,79,1
 1100 PRINT "B E E B U G's":PRINT"outline fonts":PRINT'"     by":PRINT"W. van Schaik"
 1110 VDU 28,66,31,79,30
 1120 xor=18:yor=18:sc=18:yba=15
 1130 xpos=0:xwid=55:ypos=yba:xsym=0:ysym=0
 1140 hair%=FALSE:jump%=FALSE:s%=2
 1150 ENDPROC
 1160 :
 1170 DEF PROCmenu
 1180 PRINT "New letter":PRINT "Line":PRINT "Spline":PRINT "Point":PRINT "Jump":PRINT "End of letter"
 1190 PRINT '"Fill":PRINT"X-symm. axis":PRINT "Y-symm. axis"
 1200 PRINT '"Origin":PRINT "Top":PRINT "Width":PRINT "Base"
 1210 PRINT '"Cursor":PRINT "Grid"
 1220 PRINT '"Quit"
 1230 ENDPROC
 1240 :
 1250 DEF PROCnew
 1260 *FX21,0
 1270 REPEAT:INPUT "new letter? " data$(0):UNTIL data$(0)<>""
 1280 PRINT "drawing the ";data$(0)
 1290 xsym=0:ysym=0:PROCsym
 1300 s%=2
 1310 ENDPROC
 1320 :
 1330 DEF PROCsym
 1340 data$(1)=CHR$(48+xsym)+CHR$(48+ysym)+CHR$(48+xwid)
 1350 ENDPROC
 1360 :
 1370 DEF PROCp
 1380 IF s%<3 ENDPROC
 1390 IF jump% jump%=FALSE:ENDPROC
 1400 data$(s%)=data$(s%)+CHR$(48+xpos)+CHR$(48+ypos)
 1410 PROCc
 1420 S$=data$(s%):L%=(LEN(S$))/2:IF L%>1 PROCline
 1430 PROCc
 1440 PROCwait
 1450 ENDPROC
 1460 :
 1470 DEF PROCend
 1480 PROCwrite
 1490 CLG:PROCdraw
 1500 PRINT "press key ...":A=GET
 1510 FOR i%=0 TO s%:data$(i%)="":NEXT:s
```

```
%=2
 1520 CLG:CLS:PROCfc
 1530 ENDPROC
 1540 :
 1550 DEF PROCwait
 1560 t%=TIME+30:REPEAT UNTIL TIME>t%
 1570 *FX21,0
 1580 ENDPROC
 1590 :
 1600 DEF PROCfc:PROCf:PROCc:ENDPROC
 1610 DEF PROCcf:PROCc:PROCf:ENDPROC
 1620 :
 1630 DEF PROCf
 1640 MOVE FNx(0),FNy(0):PLOT 6,FNx(0),F
Ny(55):PLOT 6,FNx(xwid),FNy(55):PLOT 6,F
Nx(xwid),FNy(0):PLOT 6,FNx(0),FNy(0)
 1650 MOVE FNx(0),FNy(yba):PLOT 6,FNx(xw
id),FNy(yba)
 1660 IF xsym<>0 MOVE FNx(xsym),FNy(0):P
LOT 6,FNx(xsym),FNy(55)
 1670 IF ysym<>0 MOVE FNx(0),FNy(ysym):P
LOT 6,FNx(xwid),FNy(ysym)
 1680 S$=data$(2):IF LEN(S$)=0 ENDPROC
 1690 FOR i%=1 TO LEN(S$) STEP 2
 1700 MOVE FNx(FNt(i%))+4,FNy(FNt(i%+1))
+4
 1710 PLOT 2,0,-8:PLOT 2,-8,0:PLOT 2,0,8
:PLOT 2,8,0
 1720 MOVE -4,-4
 1730 NEXT
 1740 ENDPROC
 1750 :
 1760 DEF PROCc
 1770 MOVE FNx(xpos),FNy(ypos)
 1780 IF NOT hair% MOVE FNx(xpos)-16,FNy
(ypos):PLOT 2,32,0:PLOT 0,-16,-16:PLOT 2
,0,32
 1790 IF hair% MOVE 0,FNy(ypos):PLOT 2,1
023,0:MOVE FNx(xpos),0:PLOT 2,0,1023
 1800 MOVE FNx(xpos),FNy(ypos)
 1810 ENDPROC
 1820 :
 1830 DEF PROCgrid
 1840 IF INKEY(-1) step%=5 ELSE step%=1
 1850 FOR h%=0 TO xwid STEP step%
 1860 FOR v%=0 TO 55 STEP step%
 1870 PLOT 70,FNx(h%),FNy(v%)
 1880 NEXT
 1890 NEXT
 1900 ENDPROC
 1910 :
```

```
 1920 DEF PROCmove(xinc,yinc)
 1930 PROCc
 1940 IF INKEY(-1) xinc=xinc*5:yinc=yinc
*5
 1950 IF FNx(xpos+xinc)<18 OR FNx(xpos+x
inc)>1012 xinc=0
 1960 IF FNy(ypos+yinc)<18 OR FNy(ypos+y
inc)>1012 yinc=0
 1970 xpos=xpos+xinc
 1980 ypos=ypos+yinc
 1990 PROCc
 2000 ENDPROC
 2010 :
 2020 DEF PROCopen
 2030 f$="FONT"
 2040 f%=OPENIN(f$):CLOSE#(f%)
 2050 IF f%=0 f%=OPENOUT(f$) ELSE f%=OPE
NUP(f$):PTR#(f%)=EXT#(f%)
 2060 ENDPROC
 2070 :
 2080 DEF PROCclose
 2090 CLOSE#(f%)
 2100 ENDPROC
 2110 :
 2120 DEF PROCwrite
 2130 d$=STR$(10000+ASC(data$(0)))+"DATA
"
 2140 FOR i%=1 TO s%
 2150 d$=d$+data$(i%)
 2160 IF i%<s% d$=d$+","
 2170 NEXT i%
 2180 FOR i%=1 TO LEN(d$)
 2190 BPUT# f%,ASC(MID$(d$,i%,1))
 2200 NEXT i%
 2210 BPUT# f%,13
 2220 ENDPROC
 2230 :
 2240 DEF PROCdraw
 2250 fill$=data$(2)
 2260 S%=1
 2270 REPEAT S%=S%+2
 2280 N$=data$(S%):S$=data$(S%+1)
 2290 L%=(LEN(S$))/2
 2300 IF N$="S" PROCspline ELSE IF N$="L
" PROCline
 2310 UNTIL N$="E"
 2320 PROCfill
 2330 ENDPROC
 2340 :
 2350 DEF PROCline
 2360 PROCline1(0,1,0,1)
 2370 IF xsym<>0 PROCline1(2*xsym,-1,0,1
)
 2380 IF ysym<>0 PROCline1(0,1,2*ysym,-1
)
 2390 IF xsym<>0 AND ysym<>0 PROCline1(2
*xsym,-1,2*ysym,-1)
 2400 ENDPROC
 2410 :
 2420 DEF PROCline1(xc,xm,yc,ym)
 2430 x=FNd(1,xc,xm):y=FNd(2,yc,ym)
 2440 MOVE FNx(x),FNy(y)
 2450 FOR N%=1 TO L%-1
 2460 x=FNd(1+2*N%,xc,xm)
 2470 y=FNd(2*(N%+1),yc,ym)
 2480 DRAW FNx(x),FNy(y)
 2490 NEXT N%
 2500 ENDPROC
 2510 :
 2520 DEF PROCspline
 2530 PROCspline1(0,1,0,1)
 2540 IF xsym<>0 PROCspline1(2*xsym,-1,0
,1)
 2550 IF ysym<>0 PROCspline1(0,1,2*ysym,
-1)
 2560 IF xsym<>0 AND ysym<>0 PROCspline1
(2*xsym,-1,2*ysym,-1)
 2570 ENDPROC
 2580 :
 2590 DEF PROCspline1(xc,xm,yc,ym)
 2600 FOR N%=0 TO L%-1
 2610 x=FNd(1+2*N%,xc,xm):X%(N%+1)=FNx(x
)
 2620 y=FNd(2*(N%+1),yc,ym):Y%(N%+1)=FNy
(y)
 2630 NEXT N%
 2640 FOR J%=1 TO L%
 2650 apeX%(J%)=((X%(J%)*4)-X%(J%+1)-X%(
J%-1))/2
 2660 apeY%(J%)=((Y%(J%)*4)-Y%(J%+1)-Y%(
J%-1))/2
 2670 NEXT
 2680 MOVE X%(1),Y%(1)
 2690 FOR J%=1 TO L%-2
 2700 FOR n=0 TO 0.5 STEP 0.02
 2710 N=1-n:m=n*2:M=1-m
 2720 IF J%=1 PROCsimplebow(J%)
 2730 IF J%>1 PROChalfbow
 2740 NEXT:NEXT
 2750 FOR n=0.5 TO 1 STEP 0.02
 2760 PROCsimplebow(J%-1)
 2770 NEXT n
```

```
 2780 DRAW FNrx(X%(J%+1)),FNry(Y%(J%+1))
 2790 ENDPROC
 2800 :
 2810 DEF PROCsimplebow(J%)
 2820 DRAW FNrx(FNbowX(J%,n)),FNry(FNbow
Y(J%,n))
 2830 ENDPROC
 2840 :
 2850 DEF PROChalfbow
 2860 LOCAL BowX,BowX2,BowY,BowY2
 2870 BowX=m*FNbowX(J%,n)
 2880 BowX2=M*FNbowX(J%-1,n+0.5)
 2890 BowY=m*FNbowY(J%,n)
 2900 BowY2=M*FNbowY(J%-1,n+0.5)
 2910 DRAW FNrx(BowX+BowX2),FNry(BowY+Bo
wY2)
 2920 ENDPROC
 2930 :
 2940 DEF FNbowX(J%,n)
 2950 LOCAL X%,X2%,apeX%
 2960 X%=X%(J%):X2%=X%(J%+2)
 2970 apeX1%=apeX%(J%+1)
 2980 =((X%+(n*(apeX1%-X%)))*(1-n))+((ap
eX1%+(n*(X2%-apeX1%)))*n)
 2990 :
 3000 DEF FNbowY(J%,n)
 3010 LOCAL Y%,Y2%,apeY%
 3020 Y%=Y%(J%):Y2%=Y%(J%+2)
 3030 apeY1%=apeY%(J%+1)
 3040 =((Y%+(n*(apeY1%-Y%)))*(1-n))+((ap
eY1%+(n*(Y2%-apeY1%)))*n)
 3050 :
 3060 DEF FNx(pos) =sc*pos+xor
 3070 DEF FNy(pos) =sc*pos+yor
 3080 DEF FNrx(x):IF x<0 =0
 3090 IF x>1023 =1023 ELSE =x
 3100 DEF FNry(y):IF y<0 =0
 3110 IF y>1023 =1023 ELSE =y
 3120 :
 3130 DEF FNt(P%) =ASC(MID$(S$,P%,1))-48
 3140 DEF FNd(P%,c,m) =c+m*FNt(P%)
 3150 :
 3160 DEF PROCfill
 3170 S$=fill$:FOR F%=1 TO LENS$ STEP 2
 3180 fx%=FNx(FNt(F%))
 3190 fy%=FNy(FNt(F%+1))
 3200 :
 3210 PLOT 133,fx%,fy%
 3220 NEXT:ENDPROC
 3230 :
```

Mastering Edit (Part 2)

by Mike Williams

In this second article on the use of the Master's Edit program, I intend to concentrate upon the search and replace facilities available. These are initiated, as I explained last month, by f4 (for a selective search/replace) and f5 for a global version. Remember, if in doubt, test things first before trusting a global search and replace with f5. That way you can abort an operation (just press Escape) before much damage has been done.

Whichever out of f4 or f5 is used, in response to the ensuing prompt, the first set of characters represents the search, followed by a slash character '/', followed by the replacement string. So the format is:

```
<search string>/<replacement string>
```

One of the immense values of Edit is that you can search for (and replace) any of the 256 characters in the range 0 to 255 (their ASCII codes). Given the format above, that immediately poses two problems: how do you search for non-printing characters or indeed any characters which cannot be found on the standard keyboard, and if the '/' character is used to separate search and replacement strings, can that character form part of either of these two strings? There are several answers to these questions as we shall see.

SPECIAL CHARACTERS

In fact, a number of *special* characters, as well as '/' have particular meanings in Edit, and as a result have to be represented differently as part of a search or replacement string. There is also a complication, and that is that the rules are somewhat different depending on whether we are searching for a character, or using it as a replacement.

When searching, the following are all considered as special characters:

```
* ^ \ / @ # . | $ - ~
```

with particular meanings in a search context. If you want to include any of these characters in a search string, then it must be preceded by backslash ('\'). Thus to search, say, in a program for a variable *title$*, you would enter as the search string:

```
title\$
```

Similarly, if you were searching for the expression *2.5*(a-b)*, this would have to be specified as:

```
2\.5\*(a\-b)
```

Note, that because the backslash has this special meaning, the only way to specify a genuine backslash is by preceding it by the relevant character, the backslash itself. To search for a backslash you must therefore include:

```
\\
```

SPECIFYING CONTROL CHARACTERS

Like the backslash character, each of the other special characters has a specific meaning within Edit. For example, the '$' character is used to indicate Return (ASCII 13). Because it occurs quite commonly, the Return character is treated differently from other control characters (characters in the range 1 to 31). All other control characters are specified by a sequence:

```
|<letter>
```

equivalent to Ctrl-<letter>. Thus the Tab character which has ASCII code 9 is therefore Ctrl-I (because 'I' is the 9th letter of the alphabet) which would be represented in a search string as:

```
|I
```

In most modes, such control characters appear on screen in an inverse video format (normally black on white in contrast to the standard white on black).

We already have the ammunition to perform some useful search and replace operations. For example, suppose we have a text file in which each paragraph ends with two Return characters, and each new paragraph starts with a single space. We want to ensure each paragraph starts with Tab. We can perform the change with:

```
$$ /$$|I
```

i.e. replace <Return><Return><Space> with <Return><Return><Tab>. We need to search for a double Return to ensure that we don't pick out other situations involving a single Return followed by a space.

TOP BIT SET CHARACTERS

As well as Control characters in the range 1 to 31, the other characters which cannot be entered directly from the keyboard are those with ASCII codes in excess of 128 (known as the top bit set characters because in a binary format the top bit, of eight, is set to '1' ensuring the code for each such character is 128 + n where 'n' is in the range 1 to 127). In Edit, these are specified by:

```
|!<char>
```

where '|!' represents ASCII code 128, and the one or two character string, <char> which follows represents a value which is added on to 128 to form the final result. Thus the character with ASCII code 129 ('Å ') would be represented as:

```
|!|A
```

because, as we have already seen, '|A' represents Ctrl-A with an ASCII code of 1. Adding this to 128 gives the required value of 129. Another example would be:

```
|!a
```

which represents the character '∝ ' (ASCII code 225). In both examples, the text for this article was read into Edit, and the sequences described above used to insert the correct characters into this text (in place of dummy characters) - View which I normally use cannot really cope with such requirements.

However, do not fall into the trap of assuming that the foregoing is only of interest if a text file contains characters in the ASCII range 129 to 255. The ability to specify these characters has other uses. For example, in Wordwise (or Wordwise Plus) Tab is represented by adding 128 to the normal ASCII code for Tab (9). Thus all Tabs are represented by ASCII 137. If you want to transfer such a file for use in View or other word processors where Tab is represented by ASCII 9 (Ctrl-I) then Edit can be used to make a global change. The search and replace specification would be:

```
|!|I/|I
```

where '|!|I' represents ASCII 137 (128 + 9), and |I of course represents Tab (Ctrl-I).

SPECIAL CHARACTERS IN REPLACEMENT STRINGS

When it comes to specifying a replacement string, there are fewer options and hence fewer special characters used up by this process. Thus only the characters:

```
% & | $
```

have to be represented by:

```
\% \& \| \$
```

This inconsistency between the use of special characters in search strings and their use in replacement strings can readily cause confusion. For example, the minus (or hyphen) is another special character, but only in search strings. Suppose you want to replace all occurrences of '-1' by '-2', the format would be:

```
\-1/-2
```

The function of '-' as a special character and others will be dealt with later.

Once the '/' character has been encountered as the separator between the search string and the replacement string, any further occurrences of '/' are treated as part of the replacement string. Thus:

```
|I\\/|I/
```

would replace <Tab>\ by <Tab>/, in other words <Tab><backslash> by <Tab><slash>.

USING SPECIAL CHARACTERS

The full list of special characters is shown in table 1. Many of these allow us to take a more flexible, and thus more powerful approach to the specification of search strings.

Normally any search is not case-sensitive, but if a letter in a search string is preceded by '\' then a match will be made only with a letter of the specified case. For example, you might have a program which uses both the pseudo Basic variable *TIME* and a variable of your own *Time* (not perhaps a good idea anyway).

Suppose you decide to change your variable *Time* to *Time%* throughout the program. If you specify (in response to f4 or f5):

```
Time/Time\%
```

then not only will Time be changed, but so will TIME (and any other combination of upper and lower case characters). Note the backslash preceding '%' in the replacement string because in this context '%' on its own is a special character with another meaning (see next month). The backslash overrules this. To achieve the correct result you will need:

```
\T\i\m\e/Time\%
```

Edit never attempts to preserve case, so the replacement string will always be used as specified.

Other specifications which can be useful include '@' which will match any single letter of the alphabet, and '#' which will match any single digit (range 0 to 9). For example, a search string of '####' would match all four digit numbers in a program listing (including line numbers and other values). One further specification for search strings which can be useful is to give a range. Thus:

```
A-F
```

would match any single character within the range 'A' to 'F' inclusive, while:

```
A-FA-F
```

would match two characters in the same range.

Char.	Meaning
*	as few of as possible
^	as many of as possible
\	treat next character as is
/	search/replacement separator
@	any letter
#	any digit
.	any character
|	next character is control character
$	Return character
-	range of characters
~	negates a specification

Table 1. Special characters used by Edit in search strings

That's really all I have time for this month. Next time in the concluding article in this short series, I will discuss how we can search for variable numbers of characters and do other clever things. For example, suppose you have a program listing in which an unwanted space appears between most line numbers and the start of the first instruction on each line. What search and replace specification would correctly remove all (or nearly all) of the unwanted spaces. This is quite tricky and needs some more of the special characters as we shall see.

In the meantime, if you want more detailed information on Edit, the place to look is Part Two of the Reference Manual for the Master series (£14.95 plus £2.50 p&p from BEEBUG or other dealers).

An Improved Route Planner

by Alan and Philip Prior

The BEEBUG Route Planner (Vol.7 No.6, November 1988) has proved very useful both for business and pleasure motoring, with the accuracy of the journey timing coping even with the vagaries of both the Dartford Tunnel and the M25. Because we used the planner so much, it became a problem to remember which filename represented which route, and so the extensions to the program described below were added, enabling a route to be selected from several on a disc by displaying starting and finishing points on a menu. With rapidly changing fuel costs, the option to change the price per gallon is added, as is the option to vary the starting time from that previously filed. All instructions for the use of the modified program are displayed on the screen.

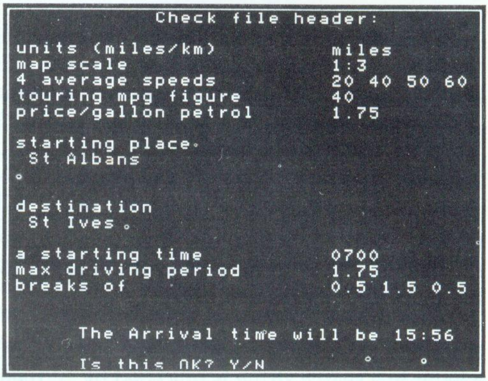
There may be cases where two routes between the same points are needed, perhaps for different times of the day or known traffic conditions. If a note is added to the destination line in the route file this will show up on the menu when using the modified program. For example, a route from Southend on Sea to Basingstoke can go either north or south of London using the M25. In this case addition of a memory jogger (e.g. "via M25N") to the destination will enable rapid identification.

UPDATING THE ORIGINAL PROGRAM

To effect the update a number of lines will have to be changed, deleted or added. It is assumed that the correction given in the March 1989 issue of BEEBUG (Vol.7 No.9) has already been incorporated. Delete lines 1520,1530,1540,1550, amend the lines shown in listing 1, and add the whole of listing 2.

As written, the modifications are for a double-sided 80 track disc using the Acorn DFS. For a single-sided disc alter the following lines:

```
3520 R%=0:buffer%=&7000:dr%=0
3670 CLOSE#0:C%=C%+8:NEXT A%
```



```
         Check file header:

units (miles/km)            miles
map scale                   1:3
4 average speeds            20 40 50 60
touring mpg figure          40
price/gallon petrol         1.75

starting place.
 St Albans
.

destination
 St Ives.

a starting time             0700
max driving period          1.75
breaks of                   0.5 1.5 0.5

     The Arrival time will be 15:56

     Is this OK? Y/N
```

Screen display produced by the Route Planner program

NEW PROCEDURES

The following new procedures are included in the program:

PROCcat
reads the directories of both disc surfaces, checks each file for a valid route and stores filename, starting and finishing points in an array.

PROCbegend
reads starting and finishing points of a route.

PROCroutename
displays a menu with available routes for selection.

PROCarrival
calculates arrival time and permits change of starting time.

PROCvdu
permits journey time calculation without route printing.

The complete Route Planner, incorporating the amendments listed here, is supplied on this month's *magazine disc. Please note that the program will not work correctly with a networked printer.*

Back issues of Vol.7 No.6 are available at the special price of £1.00 including postage. Full details on how to use the program are given in the article in that issue.

Listing 1

```
 1030 t1=5:t2=35:t3=70:t4=78:t5=89:t6=99
:t7=109:t8=120:PF%=0:ST%=0:FL%=0
 1770 startime$=FNgetstring: IF ST%=1 TH
EN startime$=startime1$
 1800 PRINTTAB(4,19)CHR$134"Is petrol pr
ice correct? Y/N" :REPEAT:V$=GET$:UNTIL
INSTR("YyNn",V$)
 1810 PRINTTAB(4,19)SPC(35):PRINTTAB(5,2
1)SPC(34):PRINTTAB(0,6)SPC30:PRINTTAB(0,
6)"price/gallon petrol"TAB(25,6);pprice:
@%=10
 1830 IF FL%=1 THEN PROCtestbuffer:CLS:P
ROCcentre("Printing Route.",12):VDU2,1,1
5,21,1,14:PRINTTAB(29)"ROUTE PLAN"' ELSE
 1890
 2050 PROCvdu(2,21)
 2270 PROCroutename
 2390 PROCvdu(2,21)
 2480 PROCvdu(2,21)
 2580 PROCvdu(2,21)
```

Listing 2

```
  105  CLOSE#0:DIMname$(62),beg$(62),end
$(62)
  125  PROCfilename
  135  PROCarrival
 1265  IF FL%=0 THEN ENDPROC ELSE VDU6
 1555  PROCcentre("Press any key to cont
inue.",20):key=GET
 1585  IF FL%=1 VDU21
 1705 IF PF%=1 THEN pprice1=pprice
 1715 IF PF%=1 THEN pprice=pprice1
 1765  IF ST%=1 THEN startime1$=startime
$
 1775 sth%=VAL(LEFT$(startime$,2)):stm%=
VAL(RIGHT$(startime$,2))
 1795  IF PF%=1 THEN 1830
 1805  IF INSTR("Nn",V$)INPUTTAB(5,21)"E
nter new price:"nprice:pprice=nprice:@%=
&20210:PF%=1
 1815  IF FL%=0 THEN PRINTTAB(5,23)CHR$1
36CHR$131"Calculating Journey Time."
 2800  DEFPROCroutename
 2810  PRINTCHR$(131)"No.";TAB(5)"Start"
;TAB(19)"Finish"
 2820  PROCcentre(CHR$(134)+CHR$(136)+"P
lease wait - reading routes.",12)
 2830  PROCcat
 2840  IF R%>16 PRINTTAB(0,21)CHR$(134)"
Press S to select route number or"'CHR$(
134)"C to continue scrolling.";SPC(10)
 2850  VDU28,0,18,39,2:CLS:r%=-1
 2860  r1%=r%:REPEAT:r%=r%+1
 2870  PRINT STR$(r%+1);TAB(5)beg$(r%);T
AB(19)end$(r%)
 2880  UNTIL r%=r1%+16 OR r%+1>=R%:IF r%
+1>=R% THEN 2920
 2890  IF R%>16 THEN REPEAT:A$=GET$:UNTI
L INSTR("CcSs",A$) ELSE 2910
 2900  IF INSTR("Ss",A$) THEN 2920
 2910  IF r%+1>=R% THEN 2920 ELSE 2860
 2920  VDU26:VDU15:PRINTTAB(0,21)SPC(46)
:PRINTTAB(5,22)CHR$(131)"Select Number o
f Route :";SPC(10)
 2930  INPUTTAB(30,22)select%
 2940  IF select%>r%+1 THEN 2840
 2950  IF select%<1 OR select%>R% THEN 2
930
 2960  f$=name$(select%-1)
 2970  ENDPROC
 2980  :
 3000  DEFPROCvdu(v1%,v2%)
 3010  IF FL%=1 THEN VDUv1%,v2% ELSE VDU
21
 3020  ENDPROC
 3030  :
 3200  DEFPROCarrival
 3210  VDU6,3: PRINTTAB(4,20)CHR$131"The
 Arrival time will be ";h%":"m%:PRINTTAB
(1,23)SPC(39);
 3220  PRINTTAB(4,22)CHR$134"Is this OK?
 Y/N":REPEAT:A$=GET$:UNTIL INSTR("YyNn",
A$):IF INSTR("Yy",A$) THEN 3250
 3230  ST%=1:INPUTTAB(4,22)"Enter New St
art Time (0000)";startime$
 3240  sth%=VAL(LEFT$(startime$,2)):stm%
=VAL(RIGHT$(startime$,2)):IF sth%>23 OR
```

Continued on page 52

1st course

Storing Data in Files

By Mike Williams

Last month's article in this series discussed some of the ways in which data can be stored and read by a program. The purpose was to look particularly at situations where a program needs access to data of some kind, but where the program itself is not intended to modify the data in any way (for example, an adventure game which needs to access a lot of text). To this end we looked at the use of DATA statements, and the associated RESTORE instruction. We also considered how a word processor or text editor might be used to create (and indeed edit) data held in a simple text format.

This month I thought it would not be out of place to take a look at some simple file handling in Basic which enables a program to both read and write data, and hence update any data previously written. Our recently published book, *File Handling for All*, provides a thoroughly readable introduction to the subject starting from first principles, but I thought I would use our First Course series to whet your appetite a little, and to provide some practical advice on simple file handling.

Handling data is, after all, a fundamental function of any computer program, and the way in which data can be structured, organised, sorted and processed has filled many a text book. At its simplest we just need to find a way of storing data in a file on disc during those periods of time that our program is not being used. In this way the data is always ready for the next time that you want to use it.

Once we start running the file program, the data will be read from the data file into the computer's memory where it is accessible to the program. The data can then be displayed or printed, it can be re-ordered, and most importantly updated with new information. Updating involves three separate activities, adding new information to that already known, changing existing information, and deleting some or any of the existing information.

When data is in memory, the simplest way of storing and accessing it is by using one or more arrays. Thus when we run our program, it will read data from a file into the arrays we have defined. Then we will be able to choose what we do with that data. Once we have finished processing, the data can be read from the arrays back to the file to wait until the next time.

Let's look at the simplest possible situation. Your program is going to use a single array; the contents of the file will be read into the array at the start of a session, and at the end the array contents will again be stored in a file. There is one flaw to this simple scenario, in that the first time you run your program, no data file exists. We just have to treat this as a special case.

Right, let's start writing some code. First of all here's an outline of the main program:

```
100 MODE 3
110 PROCinit
120 PROCtitle
130 PROCwindow
140 old=FNask("New file","Yy")
150 IF old THEN PROCread_data
160 PROCprocess_data
170 PROCwrite_data
180 END
```

The procedure PROCinit would carry out any initialisation needed by the program, and we will assume it dimensions our data array. Let's suppose this is:

```
DIM data(500)
```

That's a numeric array; it could just as easily be a string array, just as it could be a two dimensional array if that suited our needs. PROCtitle displays a title for our program on the screen, and PROCwindow defines a text window below the title which thus remains on the screen during processing.

Line 140 is used to check whether this is to be a new file, to be created for the first time, or whether we want to start by reading data from an existing file. I assume that if an existing file is to be used then the variable *old* is set to TRUE, and PROCread_data called to read the data from the file into the array. Similarly, once any processing is complete, the current data is written back to the file. Let us look at how these two vital procedures might be coded, starting first with that to write data in a file.

```
1000 DEF PROCwrite_data
1010 F=OPENOUT("MyFile")
1020 FOR I=1 TO 500
1030 PRINT#F,data(I)
1040 NEXT I
1050 CLOSE#F
1060 ENDPROC
```

That's all there is to it. However, there are quite a few assumptions, which you might wish to change. The program specifies the name of the file to hold the data as *MyFile*. You can obviously change this to whatever you like. It might be even better to allow the file name to be input at the time the program is run:

```
1005 INPUT"Enter file name: " fname$
1010 F=OPENOUT(fname$)
```

The instruction OPENOUT opens a file with the name specified ready for output. If no file of that name exists one is created automatically. If a file with that name already exists it is first deleted, and a new one created. When a file is opened, it is linked to the program via a *channel number*, which here is assigned to the variable F. All future references to the file will be by this channel number, rather than by its name. You can see this in line 1030 where a variation on the PRINT statement includes the channel number in order to print (i.e. output) data to the file.

Since our array was dimensioned to hold 500 values, the FOR-NEXT loop write each array element in turn out to the file. Once this process has been completed, the program should close the file, which it does using the CLOSE statement and again references the file through its channel number. The procedure then terminates.

To read data from a data file involves a very similar routine.

```
1100 DEF PROCread_data
1110 F=OPENIN("MyFile")
1120 FOR I=1 TO 500
1130 INPUT#F,data(I)
1140 NEXT I
1150 CLOSE#F
1160 ENDPROC
```

Almost the only difference is in the use of OPENIN to open the data file. The instruction OPENOUT will delete any existing file with the name specified before creating a new empty file. That's no good when we want to read in the data. This is where OPENIN comes in; it opens an existing file ready for use.

Again we could make the routine more flexible by allowing the file name to be specified when the program is run. This means that we could easily store more than one set of data using that which is appropriate to the task in hand. For example, we might have several files each storing the average temperature per month for a different place.

Another useful variation is to include a check that the file specified does in fact exist (we might just have mis-typed its name). Just add the line:

```
1115 IF F%=0 PRINT"No such file":ENDPROC
```

A channel number of zero indicates that the file specified could not be found.

Well, as far as file handling goes that's about it. Follow the pattern shown and you should be able to do just what we set out to do: use a data file to hold the contents of an array while a program is not being used (and your computer is switched off).

And the routines could easily be extended. For example, there is no necessity to restrict the read and write routines to a single array. As many arrays as you like could be stored in the one file - just add more FOR-NEXT loop routines to each of the two procedures. Do make sure that in such a case you both read and write the arrays in the same order. Likewise, although both numeric and string arrays could be stored in the same data file, make sure that everything corresponds, i.e. you read numeric data into a numeric array, and string data into a string array.

Another tip which might save time is to use the zero element in each array to store the number of elements currently in use. You might have dimensioned the array to hold 500 values, but maybe only the first 200 of these are being used. When writing the data to an array you will need to include an extra line, and modify the start of the FOR loop:

```
1015 PRINT#F%,data(0)
1020 FOR I%=1 TO data(0)
```

Similarly, when reading data from the file into the array, add one line, and amend another:

```
1115 INPUT#F%,data(0)
1120 FOR I%=1 TO data(0)
```

Again, the same technique can be applied if more than one array is to be stored in the same data file, by preceding each read or write loop with an instruction like that at line 1015 (write) or at 1115 (read).

Of course, if you are dealing with a string array this adds a slight complication, but this is easily overcome by converting the number of elements to a string before writing to the file:

```
1015 PRINT#F%,STR$(data(0))
```

or converting from a string back to numeric form when reading:

```
1115 INPUT#F%,n$:data(0)=VAL(n$)
```

Of course, there is a whole lot more to file handling than the little I have covered here, but I hope I have managed to show you just how simple file handling can be, something that's worth remembering when you start getting involved in more advanced file handling techniques. And yet the ideas I have covered here have definite practical applications.

There is also, of course, one significant limitation in the approach used here, and that is that the amount of data which you can store in a data file using the techniques described here is limited by the amount of available memory for the arrays you use - in mode 3, say, on a simple model B with a program of moderate length that might not be very much. The facility for a program to be able to create and process data files much larger than the available memory is certainly one of the reasons for using more sophisticated routines.

Another consideration, particularly with large files, is the speed with which various tasks can be carried out - search for a particular piece of information, or ordering the data in a file in some way. As I hope you can see there is a lot more to file handling, and hence the need for our own book on the subject which I mentioned at the outset. At the same time I hope you have learnt something of interest from even the simple approach taken here.

Next month, as they say, I hope to move on to something completely different.

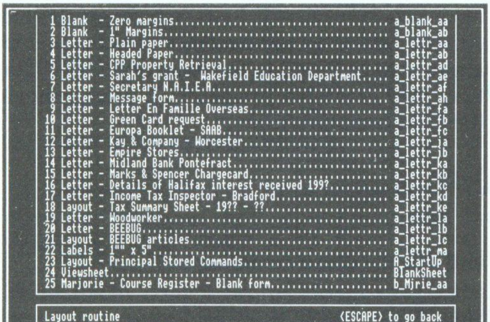
An ADFS/View Utility for the Master (Part 1)

Optimise your use of the View word processor with Jeff Gorman's masterly utility.

INTRODUCTION

The DFS offers a straightforward approach to keeping word processing files, but there can be difficulties where most of the work is concerned with short files such as business letters. The limit of 31 files per disc face can be reached well before the disc is full. If one has quite a number of standard document layouts to store, there is even less space for printable texts.

The ADFS on the other hand, while more complicated to use, overcomes the 31 files/disc limitation of the DFS. The program *PreView* (part one this month, part two the next) helps make the ADFS painless to use, allowing a notional 2162 text files, although taking things that far would mean a limit of about 30 printable words per file. The program also improves the convenience of using View. Within certain limitations, PreView can also recognise and handle ViewSheet files.



```
 1 Blank  - Zero margins........................................ a_blank_aa
 2 Blank  - 1" Margins.......................................... a_blank_ab
 3 Letter - Plain paper......................................... a_lettr_aa
 4 Letter - Headed Paper........................................ a_lettr_ab
 5 Letter - CPP Property Retrieval.............................. a_lettr_ad
 6 Letter - Sarah's grant -  Wakefield Education Department..... a_lettr_ae
 7 Letter - Secretary N.A.I.E.A................................. a_lettr_af
 8 Letter - Message form........................................ a_lettr_ah
 9 Letter - Letter En Famille Overseas.......................... a_lettr_fa
10 Letter - Green Card request.................................. a_lettr_fb
11 Letter - Europa Booklet - SAAB............................... a_lettr_fc
12 Letter - Kay & Company - Worcester........................... a_lettr_ja
13 Letter - Empire Stores....................................... a_lettr_jb
14 Letter - Midland Bank Pontefract............................. a_lettr_ka
15 Letter - Marks & Spencer Chargecard.......................... a_lettr_kb
16 Letter - Details of Halifax interest received 199?........... a_lettr_kc
17 Letter - Income Tax Inspector - Bradford..................... a_lettr_kd
18 Layout - Tax Summary Sheet - 19?? - ??....................... a_lettr_ke
19 Letter - Woodworker.......................................... a_lettr_la
20 Letter - BEEBUG.............................................. a_lettr_lb
21 Layout - BEEBUG articles..................................... a_lettr_lc
22 Labels - 1"" x 5"............................................ a_lettr_ma
23 Layout - Principal Stored Commands........................... A_StartUp
24 Viewsheet.................................................... BlankSheet
25 Marjorie - Course Register - Blank form...................... b_Mjrie_aa

Layout routine                                           <ESCAPE> to go back

Key index (1 to 25) or "M" for more & <RETURN>
```

Preview's list of folders and files

This month's listing creates an operational, but scarcely friendly program. The screen design is perhaps best described as spartan. The second part will improve the presentation, extend the program to allow greater tolerance of human frailty, and allow recovery from a false move. Also to be included is a short program "MultiPt" which, when used in conjunction with PreView, offers multiple automated printouts of a file from a command issued by a function key.

FEATURES OF PREVIEW

View needs main memory for its own purposes, so on switching from Basic to View, any existing program is overwritten. To enable almost instantaneous switching from View to PreView, the program is stored in sideways RAM ready for re-loading back into main memory. This is much quicker than re-loading from disc, making this kind of program vastly more convenient to use.

PreView guides the user through the stages needed to select from a descriptive list of his/her own standardised layouts. Once a file has been selected, it configures View in mode 3 using shadow memory, with the selected layout given a verified filename, in *Edit* mode, with character entry in *Insert* mode and the cursor in place for creating a one line description of the text to be composed (see below). It is theoretically possible, but not advisable, to organise 47 View files into each of 46 *Folders*. A Folder is a sub-directory containing a set of *Texts* (files), or in the special case of *Folder.Layout*, layouts only.

Provided the user includes a stored command (namely "CO", using Shift-f8), immediately beneath the top ruler of each text, and enters on the same line, a one-line description of the contents of the file, then the standard ten letter file name for texts can effectively be extended to a sixty-two character description plus the filename. This should greatly ease the familiar problem of remembering, after a period of non-use, the contents of files if prompted only by more or less cryptic ten letter filenames. Non-View files can also be identified.

USING THE FUNCTION KEYS WITH PREVIEW

From View's command mode, function key f0 now offers a single key press for saving to drive 0. If initialised for dual drives, key f1 saves to drive 1 and returns the user to drive 0 in the correct directory. With dual drives, backups can therefore be readily generated as one works. It is quite possible to automate saves to two drives with one key press, but I have found the present system to be more blunder resistant, and therefore prefer to "waste" a function key. Other function keys are defined as follows.

F2 calls the printer driver (see the note below).

F3 calls a program offering printing of multiple copies of the text currently in use (full version only).

F4 recalls "PreView" for almost instant re-use (from sideways RAM).

From View's Edit mode, **Shift-Ctrl-f5** generates at the cursor position, a reference code based on folder and filenames, provided that one is working from Option 1.

Shift-Ctrl-f6 can be programmed to give, at the cursor position, a permanent entry of the current date, given that an already published program is added (see below for details).

Shift-Ctrl-f7 extends View to offer "DELETE WORD" if the cursor lies on or between the space before a word and the last character, albeit not if this is the last word on a line.

Shift-Ctrl-f8 gives "Delete up to cursor" (i.e. from the beginning of the line).

Shift-Ctrl-Copy puts a *Highlight 1* at the beginning of the line and at the cursor position - useful for underlining section headings and the like.

At start-up, PreView shows the disc capacity used, and offers an estimate of the remaining number of A4 sheets, based on a rough estimate of 4000 characters per sheet. There is a warning if the disc is more than 95% full and disc management is advisable.

The layout list is always revised if it appears to PreView that the Layout folder has been accessed via the Edit option, and therefore assumed to have been edited. Presumably this will not happen very often, and the consequent delay therefore tolerable. To avoid unnecessary delays during normal working, other folder lists are updated only if the program detects a change in the numbers of texts, so modified comment lines will not be shown until the disc is re-booted. If folders are created or deleted, remember to re-boot the disc, rather than recall PreView with f4.

Memory in the regions of &900 and &A00 might be needed for other favourite machine code utilities which extend View. To preserve these areas, some of the memory used by Econet and memory in page 13 is used to retain information while skipping from Basic to View.

INSTALLING PREVIEW

Start with an empty disc. Since the second instalment interposes line numbers within the first program, as well as overwriting others and adding a number of additional procedures and functions, take care when entering the program to type the line numbers exactly as they are printed. Save the program as *$.PreView_1*. If just a single drive is available, amend line 1040, as indicated in the listing.

Note that the utility (FNdate) which enables the current date to be permanently incorporated in a text, is to be found in the issue of BEEBUG for May 1989 (Vol.8 No.1 page 29). If used, remove the REM from line 3100.

Type:

```
*Dir:0
```

and title the disc by entering:

```
*Title "WP Files-Personal"
```

or whatever title is preferred.

You will also need to create the following (essential) !Boot file as shown:

```
*Build $.!Boot
1 *Basic
2 ?&D90=1:?&D96=0:$&B10=""
3 CHAIN "$.PreView_1"
```

Press Escape and enter:

```
*OPT4,3
```

To set up the disc to use PreView, continue as below:

```
*Word
*CDir$.Folder
*CDir $.Folder.Layout
*Dir  $.Folder
*CDir <Folder-name>
```

the latter entered as many times as required, but at least once, to create as many folders as needed (up to 46 in addition to the *Layout* folder). Use folder names appropriate to your own needs. Next enter:

```
*Dir  Layout
```

and then press Escape to enter Edit mode in View. Prepare the first layout which might possibly be just a series of unquantified *Edit* commands such as TM, HM, DF, PL, FM, DF, BM, LM as described in the View manual. To make full use of PreView, it is essential to have, on the line immediately below the top ruler, a stored command, "CO" (created by using Shift-f8), followed on the same line by an aide memoire describing the contents of the file.

From command mode again, type:

```
Save A_SetUp
```

and then:

```
*Dir:0
```

to return to the root ($) directory.

The program will not operate unless there is at least one layout in the Layout folder, and one other, e.g. *$.Folder.Holidays*.

If available, copy a printer driver program to the root directory and alter "$.Star" in line 3070 to the name of the printer driver, not forgetting the "$" symbol and the dot.

Use Shift-Break to run PreView and check its operation. Option 1 should now reveal one file in the layout folder which can be used to prepare other layouts for the Layout folder. The full sequence from Option 1 can be used to prepare normal texts. With care, the program works smoothly, but for the time being, expect to be vexed by mis-keyed entries, although f4 will always re-start the program. At present, the *Escape to go back* option does not operate.

Instead of directly preparing layouts from PreView, existing texts can be copied to the Layout sub-directory, and edited from Option 2. The Welcome disc, or BEEBUG's Master ROM will help with multiple copying, especially if copying from the DFS. The ADFS pathname is *$.Folder.Layout* or *$.Folder.Whatever* if copying other existing files to this system.

Note: Since the user no longer relies on filenames for identification purposes these can, without much disadvantage, become quite cryptic. The ADFS lists filenames alphabetically. If so desired, the layout list can be organised by systematic choice of names, thereby making it possible to group layouts according to function. Frequently used layouts can be brought to the top of the list.

If it is preferable for the printer driver to be loaded always, remove the REM from line 3140, but the consequent disc activity incurs some loss of speed.

```
   10 REM Program PreView_1
   20 REM Version B1.3
   30 REM Author  Jeff Gorman
   40 REM BEEBUG  March 1991
   50 REM Program subject to copyright
   60 :
  100 ON ERROR GOTO 300
  110 *Shadow
  120 MODE 128
  130 DIM txt$(47),tex% 78,fdr% 10
  140 DIM blk% 13,reply% 18
  150 *SRData4
  160 *SRData5
  170 *SRWrite E00+3500 3FFF
  180 PROCinit
  200 IF ?&D90=0 AND ?&D98=1 PROCrevLyts
  210 IF ?&D90=1 PROCchargeSWRAM
  220 PROCmenu:END
  230 :
  300 PROCcsr(1):VDU26,12,31,0,15:COLOUR
 1:*CLOSE
  310 REPORT:PRINT" at line "ERL;:END
  320 :
 1000 DEF PROCcsr(s%)
 1010 VDU23,1,s%;0;0;0;:ENDPROC
 1020 :
 1030 DEF PROCinit:*FX18
 1040 ?&D91=1:REM Dual drives else ?&D91
=0
 1060 nFld%=?&D92:nTxt%=?&D93:oldTxt%=?&
D93:nLyt%=?&D94:nItm%=0:dir$="":tp$ =STR
ING$(78,"t"):tp$="":title$=STRING$(78,"l
"):title$="":cont$="Press any key to con
tinue":wt$="Please wait"
 1080 *Key4 *Basic|M*SRRead E00+3500 3FF
F|MOLD|MRUN|M
 1090 seeFnme%=0:inRam%=0:reRun%=0:esc%=
0:slctLyt%=0:*FX202,48
 1100 ENDPROC
 1110 :
 1120 DEF PROCrevLyts
 1130 PROCmsg(28,2,"Layout folder appear
s to have been revised",0):PROCstLyts:EN
DPROC
 1140 :
 1150 DEF PROCstLyts:PROClist("Layout","
.Layout"):PROCstoreCOs("lyt"):PROCclr:?&
D98=0:ENDPROC
 1160 :
 1170 DEF PROCstoreCOs(ty$)
 1180 IF ty$="lyt" nd%=nLyt% ELSE nd%=nT
xt%
 1190 IF nd%=0 ENDPROC
 1200 FOR nm%=1 TO nd%:title$=""
 1210 IF nm%<=9 nm$=" "+STR$nm% ELSE nm$
=STR$nm%
 1220 fil$=txt$(nm%):title$=nm$+" "+FNge
tCO+" "+txt$(nm%):PROCst(title$,nm%,ty$)
 1240 NEXT:ENDPROC
 1250 :
 1260 DEF FNgetCO:LOCALasc%:F%=0:H%=OPEN
INfil$:O%=BGET#H%:P%=BGET#H%
 1270 IF EOF#H% F%=10:CLOSE#H%:="Void fi
le"+STRING$(52,".")
 1280 Q%=BGET#H%:tp$="Viewsheet":F%=27:I
F O%=128 AND P%=67 AND Q%=79 F%=0:tp$=""
:tp$=FNreadCO
 1290 CLOSE#H%:=tp$+STRING$(61-LENtp$,".
")
 1300 :
 1310 DEF FNreadCO:tp$=LEFT$(tp$,F%):REP
EAT:F%=F%+1:asc%=BGET#H%:tp$=tp$+CHR$asc
%:UNTIL F%=62 OR asc%=13:=LEFT$(tp$,LENt
p$-1)
 1320 :
 1330 DEF PROCclr:FOR B%=0 TO 47
 1340 txt$(B%)="":NEXT:ENDPROC
 1350 :
 1360 DEF PROCchargeSWRAM:*Dir:0
```

```
 1380 PROCrdDirNstFld("Folder",5):PROCli
st("Folder",""):PROCstLyts:?&D90=0:ENDPR
OC
 1390 :
 1400 DEF PROCmsg(l%,s%,msg$,cur%)
 1410 PROCcsr(cur%):VDU28,s%,l%,77,l%,12
 1420 PRINT msg$;:ENDPROC
 1430 :
 1440 DEF PROCrdDirNstFld(d$,A%):D%=0:nm
e$="":osgbpb=&FFD1:?blk%=0:blk%!1=reply%
 1450 blk%!5=1:blk%!9=0:X%=blk% MOD 256:
Y%=blk% DIV 256:REPEAT:D%=D%+1:blk%?5=1:
blk%?1=reply% MOD 256:blk%?2=reply% DIV
256:CALL osgbpb:?(reply%+1+?reply%)=13:n
me$=$(reply%+1)
 1460 IF A%=5 $&B10=nme$
 1470 IF d$="Folder" AND A%=8 AND nme$>"
" PROCst(nme$,D%,"fnms")
 1480 IF D%<=47 AND (d$="txt" OR d$="Lay
out") txt$(D%)=nme$
 1490 UNTIL blk%?5=1 OR D%=48
 1500 IF d$="Folder" nFld%=D%-1:?&D92=nF
ld%
 1510 IF d$="txt"    nTxt%=D%-1:?&D93=nT
xt%
 1520 IF d$="Layout" nLyt%=D%-1:?&D94=nL
yt%
 1530 ENDPROC
 1540 :
 1550 DEF PROCst(st$,S%,type$):PROCsetVa
rs(st$):OSCLI ("SRWRITE " + STR$~addr% +
 "+"+ STR$~(len%)+" "+STR$~(offset%+(S%*
len%))):ENDPROC
 1560 :
 1570 DEF PROCsetVars(st$)
 1580 IF type$="fnms" len%=10:offset%=0:
$fdr%=STRING$(10," "):$fdr%=st$:addr%=fd
r%:ELSE len%=78
 1590 IF type$="lyt" offset%=500
 1600 IF type$="txt" offset%=4200
 1610 IF type$="txt" OR type$="lyt" $tex
%=STRING$(78," "):$tex%=st$:addr%=tex%
 1620 ENDPROC
 1630 :
 1640 DEF PROClist(a$,b$)
 1670 OSCLI("Dir $.Folder"+b$)
 1680 PROCclr:PROCrdDirNstFld(a$,8):ENDP
ROC
 1690 :
 1700 DEF PROCmenu:lyt%=0:txt%=0:fld%=0:
useLyt%=0:dir$=""
 1720 ?&D98=0:LOCAL k$:PROCwnd:CLS:PROCs
howMnu:PROCmsg(28,2,"Disc Title:- "+$&B1
0,0)
```

```
 1740 PROCmsg(30,2,"Key option - or Esca
pe to quit",1):REPEAT:k$=CHR$ GET:UNTIL
INSTR("12",k$):IF k$="1" PROClytRtne
 1750 IF k$="2" PROCtxtRtne
 1760 ENDPROC
 1770 :
 1780 DEF PROCwnd:PROCcsr(0)
 1790 VDU28,0,26,79,0:ENDPROC
 1800 :
 1820 DEF PROCshowMnu
 1850 PRINT TAB(0,22);:REM first stage o
nly
 1860 PRINT " (1) Use layouts to start a
 new text"'" (2) Edit existing texts/la
youts";:ENDPROC
 1870 :
 1880 DEF PROClytRtne:lyt%=TRUE:useLyt%=
TRUE:seeFnme%=0:PROClytList:PROCgetFnme:
txt%=0 :PROCgetFolds:IF NOT inRam% PROCs
ize
 1910 PROCmakeFnme
 1920 PROCgetView:ENDPROC
 1930 :
 1940 DEF PROClytList:CLS
 1960 PROClistNpickTxt:ENDPROC
 1970 :
 1980 DEF PROCgetFnme:IF (txt% AND dir$<
>"Layout") OR inRam% source$="txt"
 1990 IF lyt% OR slctLyt% source$="lyt"
 2000 sln$=LEFT$(FNfetch(num%,source$),5
9)
 2010 IF inRam% OR lyt% fil$=RIGHT$(FNfe
tch(num%,source$),10) ELSE fil$=txt$(num
%)
 2015 IF useLyt% lytFil$=fil$
 2030 $&B00=fil$:ENDPROC
 2040 :
 2050 DEF FNfetch(B%,type$):PROCsetVars(
"")
 2060 OSCLI("SRREAD " + STR$~(addr%)+"+"
+STR$~(len%)+" "+STR$~(offset%+(B%*len%)
))
 2070 IF type$="lyt" OR type$="txt":=$te
x% ELSE:=$fdr%
 2080 :
 2090 DEF PROCgetFolds
 2110 fld%=TRUE:PROCfoldIndx:PROCpickFol
d
 2130 ENDPROC
 2140 :
 2150 DEF PROCfoldIndx
 2160 PROCwnd:CLS:REM remove later
 2170 IF useLyt% PROCmsg(28,2,"Classify
new text"+STRING$(38," ")+"Escape to go
back",0)
 2190 PROCprntFolds:*FX202,48
 2200 ENDPROC
 2210 :
 2220 DEF PROCpickFold:oldDir$=$&D80
 2240 oldNum%=?&D96:ind$=" (1 to "+STR$n
Fld%+")":num%=VAL(FNindex("Key index "+i
nd$,nFld%)):?&D96=num%:IF num%=0 OR num%
>nFld% VDU7:PROCpickFold:ENDPROC
 2250 dir$=LEFT$(FNfetch(num%,"fnms"),IN
STR(FNfetch(num%,"fnms")," ")-1)
 2260 inRam%=(num%=oldNum% AND dir$=oldD
ir$) OR (dir$="Layout" AND NOT esc%)
 2270 IF useLyt% OR esc% PROCsize:chkNo%
=nTxt% ELSE chkNo%=0
 2290 $&D80=dir$:IF dir$="Layout" ?&D98=
1
 2300 slctLyt%=((dir$="Layout") AND tex%
)
 2330 ref$=LEFT$(dir$,4)+".":ENDPROC
 2340 :
 2360 DEF PROCsize:IF NOT inRam% OSCLI("
Dir $.Folder."+dir$)
 2370 PROCclr:PROCrdDirNstFld("txt",8):E
NDPROC
 2380 :
 2390 DEF PROClistNpickTxt:IF txt% AND N
OT seeFnme% PROCmsg(28,2,"Select a text"
,0)
 2400 IF lyt% OR dir$="Layout" typ$="lyt
":nItm%=nLyt% ELSE typ$="txt":nItm%=nTxt
%
 2410 IF nItm%<=25 stp%=nItm% ELSE stp%=
25
 2420 IF stp%=0 stp%=1
 2430 y%=FNfixTxtBox(stp%+1):PROCwnd:CLS
:REM first stage only
 2450 begin%=TRUE:REPEAT:code$="":a$=""
 2460 IF begin% sa%=1:st%=stp% ELSE sa%=
nItm%-24:st%=nItm%
 2470 IF NOT reRun% VDU28,1,25,77,y%+1,1
2:PROCcsr(0):VDU28,1,y%+stp%,77,y%+1:PRO
CprntTxtLst(sa%,st%,typ$)
 2480 IF nItm%>25 more$="or ""M"" for mo
re" ELSE more$=""
 2490 IF nItm%=1 index$="Key 1" ELSE ind
ex$="Key index ("+STR$sa%+" to "+STR$st%
+") "+more$
 2500 IF seeFnme% index$="Key ""F"" to a
dopt a filename "+more$
 2520 code$=FNindex(index$,nItm%)
 2530 IF code$<>"F" num%=VALcode$:begin%
=NOTbegin%:reRun%=0
 2540 UNTIL code$<>"M" OR code$="F":seeF
```

```
nme%=0:IF code$="F" PROCmakeFnme
 2550 ENDPROC
 2560 :
 2580 DEF FNindex(ky$,lmt%):*FX21,0
 2590 PROCmsg(30,2,ky$+" & Return ",1):I
NPUT""i$
 2600 IF i$="f" i$="F"
 2610 IF i$="m" i$="M"
 2630 =i$
 2640 :
 2650 DEF PROCprntTxtLst(strt%,d%,typ$)
 2660 IF nItm%=0 AND lyt%:ENDPROC
 2670 IF nItm%=0:PRINT " No texts in Fol
der """dir$"""";:PROCmsg(28,2,"Return to
 the Menu",0):PROCcont(""):PROCmenu:ENDP
ROC
 2680 PRINT TAB(0,0);:FOR num%=strt% TO
d%
 2690 PRINT " "+LEFT$(FNfetch(num%,typ$)
,76);:IF num%<d% PRINT
 2700 NEXT:ENDPROC
 2710 :
 2720 DEF PROCmakeFnme:PROCmsg(30,2,"Ent
er a filename (max 10 characters) ",1)
 2740 INPUT""rep$:IF LENrep$>10 OR LENre
p$=0 VDU7:PROCmakeFnme:ENDPROC
 2760 txtFil$=rep$:ref$=ref$+rep$:ENDPRO
C
 2770 :
 2790 DEF FNfixTxtBox(nl%):IF nl%>=25:=0
 2800 =26-nl%
 2810 :
 2820 DEF PROCcont(text$):PROCmsg(30,2,t
ext$+cont$,0):G=GET:ENDPROC
 2830 :
 2840 DEF PROCprntFolds:PROCcsr(0):dir%=
0:y%=FNfixFoldBox:VDU28,1,26,78,y%,12
 2860 REPEAT:rspc$="":dir%=dir%+2:line%=
dir%/2:IF line%=1 fd$="Folder" ELSE fd$=
STRING$(6," ")
 2870 IF line%=1 head$=STRING$(2,CHR$32)
+"Name:-"+STRING$(7,CHR$32) ELSE head$=S
TRING$(2,CHR$32)+STRING$(11,CHR$166)+STR
ING$(2,CHR$32)
 2880 PROCprntFoldLine
 2890 UNTIL dir%>=nFld%:ENDPROC
 2900 :
 2910 DEF FNfixFoldBox
 2920 n%=(nFld%/2)+(nFld% MOD 2):=25-n%
 2930 :
 2940 DEF PROCprntFoldLine:lf$=FNfetch(d
ir%-1,"fnms"):rf$=FNfetch(dir%,"fnms"):I
F lf$= "" ENDPROC
 2950 PRINT SPC2 fd$;:@%=&02:PRINT dir%-
1 head$ lf$ SPC3 CHR$169 SPC2;
 2960 IF rf$="" OR dir%>=nFld%+1 ENDPROC
 2970 PRINT fd$;:@%=&02:PRINT dir% head$
 rf$ SPC(1);:IF dir%< nFld% PRINT
 2980 ENDPROC
 2990 :
 3000 DEF PROCgetView:*FX 229
 3030 PROCwnd:CLS:COLOUR0:*FX228,1
 3040 *FX 11,25
 3050 OSCLI("Key0 Save|M"):*FX200,0
 3060 IF ?&D91=1 OSCLI("Key1 *Dir:1.$.Fo
lder."+dir$+"|MSave|M*Dir:0.$.Folder."+d
ir$+"|M")
 3070 OSCLI("Key2 Printer $.Star|M*Dir$.
Folder."+dir$+"|M")
 3090 IF k$="1" OSCLI("Key5 "+ref$) ELSE
 OSCLI("Key5")
 3100 REM OSCLI("Key6 "+FNdate)
 3110 OSCLI("Key7|!|Y|!|Y|!(|!|_ ")
 3120 OSCLI("Key8|!#2|!|P|!#1|!,")
 3130 OSCLI("Key11|!|P|! |!|Q|! ")
 3140 PROCloadNine:REM *FX 138,0,130
 3150 ?&D95=1::*FX 138,0,137
 3160 IF LEFT$(dir$,9)="ViewSheet" OSCLI
("Sheet") ELSE OSCLI("Word")
 3170 ENDPROC
 3180 :
 3190 DEF PROCloadNine
 3200 lyt$="Key 9 *Dir $.Folder.Layout"
 3210 fold$="*Dir $.Folder."+dir$
 3220 S$=" |M SetUp F I|M M 3|MSearch^?|
M"
 3230 IF useLyt% OSCLI(lyt$+" |MLoad "+l
ytFil$+" |M *Dir $.Folder.|M Name "+txtF
il$+"|M*Dir $.Folder."+dir$+S$)
 3240 IF txt% OSCLI("Key9 "+fold$+" |M L
oad "+fil$+S$)
 3250 ENDPROC
 3260 :
 3270 DEF PROCtxtRtne:useLyt%=0:seeFnme%
=0
 3280 PROCgetFolds:txt%=TRUE:PROCsize:PR
OCgetTxts:PROCgetFnme:PROCgetView:ENDPRO
C
 3290 :
 3300 DEF PROCgetTxts:typ$="txt":lyt%=0
 3310 IF (inRam% AND oldTxt%=nTxt%) OR s
lctLyt% OR seeFnme% PROClistNpickTxt:END
PROC
 3320 PROCstoreCOs("txt")
 3330 IF lyt% nItm%=nLyt% ELSE nItm%=nTx
t%
 3340 PROClistNpickTxt:ENDPROC
```

Data Compression

by Bernard Hill

Last month we looked at data encryption as a means of changing the contents of a message so that it becomes undecipherable by unauthorised readers. This month we will be looking at data alteration again, but in the context of data compression in order to make it smaller so that it can be stored in less disc space or transmitted down a serial line in faster time. In large bulletin boards file compression is quite usual, but of course compression and decompression programs have to be supplied to the users.

ELEMENTARY COMPRESSION METHODS

One of the very simplest of compression techniques is Run-Length Encoding. If a character is to be repeated in the file then we replace it by a special character denoting a repetition count. Suppose our message consists of the string:

```
AAAABBBAABBBBBCCCCCC
```

If we know that the message consists of only ASCII codes below 127 then we could use ASCII 127-255 to represent repetition of the last character 3 to 131 times (ASCII 127=3 times, 128=4 times etc.). Then the above message becomes:

```
A<128>B<127>AAB<129>C<130>
```

which is a compression of 20 bytes to 10. This method is of best use in long repetitions, such as screen dumps where the screen contains large areas of uniform colour.

Another elementary method of file compression is to use codes known to be absent from the text (as ASCII 127 etc. as above) to represent common repetitive sequences, e.g. in English 'ing' is fairly common and might be represented by <127>, 'es' by <128> and so on. It is a long process to find the 128 most common 2-character and 3-character sequences in a piece of text in order to substitute these codes, but a combination of this method and Run-Length Encoding was used to shoe-horn 28K of text into less than 15K for Beebug's HELP ROM. But since the program which searched for the commonest combinations ran for about 24 hours on a 6502 co-processor it is of limited help in a general context!

VARIABLE LENGTH ENCODING

No matter how often it is used, each character in a normal text file consumes 8 bits. One of these bits is redundant in that all ASCII codes can be represented in 7 bits, so by omitting the top bit of every character before storing it we can always save 12.5% of space. We can do considerably better than this, however, in the vast majority of cases. Simply represent the commonest characters with a smaller number of bits, and the uncommon ones with a larger number. Let's illustrate this with a sample message:

```
BBC BY BEEBUG
```

We could encode B in binary as '0' since it occurs 5 times, E as '1' since it occurs twice, C as '01', Y as '10', G as '00', space as '11' and U as 000:

```
B B C     B Y     B E E B U   G
0 0 01 11 0 10 11 0 1 1 0 000 00
```

but 00011101011011000000 isn't a valid code because we need to include delimiters to ensure the code is unique (it could represent BBBEEEBEBEBBBEBBEEEEEE or many other messages). But we wouldn't need delimiters if no code is the prefix of any another. Consider this code for the same message:

B	0
E	101
space	100
C	1100
G	1101
U	1110
Y	1111

i.e.:

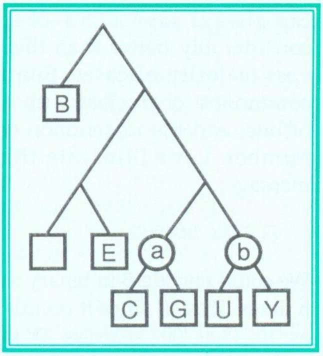
```
B B C      sp  B Y     sp  B E    E    B U     G
0 0 1100 100 0 111 100 0 101 101 0 1110 11
01
```

giving 32 bits in all:

```
00110010001111000101101011101101
```

Only 4 bytes are needed for a message of 13 bytes. Now this code is unique as you can tell if you try decoding it. The first zero must be B since no other code starts with 0. Similarly the second. There is no '1' or '11' or '110' so the next group must be '1100' or C.

An easy way to represent this code is as a tree, where a '0' means 'go left' and a 1 means 'go right' as shown here:



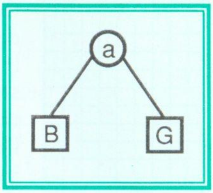
But how do we discover the best way to represent each character and so build this tree? It turns out that there is an elegant way to compute this tree giving us a bit string of minimal length for any given message, discovered by D.Huffman in 1952. It gives a compression by 44% on the text of this article, for example.

HUFFMAN ENCODING

First count the number of occurrences of each letter and sort into ascending order:

```
C  G  U  Y  spc  E  B
1  1  1  1   2   2  5
```

We are now going to build the tree show above from the bottom up: first combine the C and G to form a subtree whose node contains the sum of the occurrences of C and G:



Call this node 'a' with frequency the total of C and G, and add it to the end of the node/ frequency list:

```
C  G | U  Y  spc  E  B  a
1  1 | 1  1   2   2  5  2
```

Now omit the first two and sort the top part into order of the frequencies so that 'a' falls to its correct position:

```
C  G | U  Y  spc  E  a  B
1  1 | 1  1   2   2  2  5
```

Then merge U and Y into a new subtree 'b' of size 2, and sort the top part again:

```
C  G  U  Y |spc  E  a  b  B
1  1  1  1 | 2   2  2  2  5
```

Now combine space and E to form another subtree of size 4 called 'c' and sort:

```
C  G  U  Y  spc  E | a  b  c  B
1  1  1  1   2   2 | 2  2  4  5
```

'a' and 'b' now form another node 'd' of size 4, sorted into place:

```
C  G  U  Y  spc  E  a  b | c  d  B
1  1  1  1   2   2  2  2 | 4  4  5
```

And again, c+d=e, no movement during the sort this time:

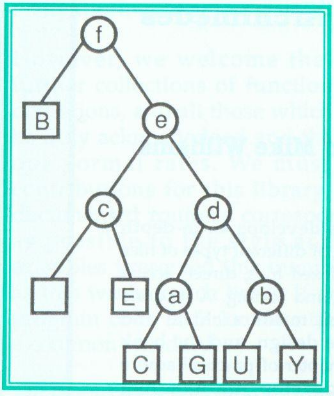
```
C  G  U  Y spc E  a  b  c  d | B  e
1  1  1  1  2  2  2  2  4  4 | 5  8
```

and lastly join B and e to form f:

```
C  G  U  Y spc E  a  b  c  d  B  e  f
1  1  1  1  2  2  2  2  4  4  5  8 13
```

Now remembering the definition of our new entries a to f:

a=C+G, b=U+Y ... f=B+e we can see the tree structure we showed above:



Listing 1 captures this algorithm from a sequence of bytes in a named file (of course the algorithm can handle program files as well as text files) where the following points should be noted:

The program forms the arrays code% and len%, one for each ascii character, thus in the above example the letter C is ASCII 67 and code%(67)=12 or 1100 in binary, while len%(67)=4, ie 4 bits. Unused letters such as A have len% set to zero (len%(65)=0).

The character counts are in total% and the tree nodes (which cannot exceed 512 in number) are represented by arrays as follows:

node%	- ascii values at the terminal nodes
count%	- frequency counts at the nodes
dad%	- the upward structure of the tree so that a negative value indicates that this node is a left son, positive a right son.

The sort is a bubble sort which is particularly fast in the case of an addition to the list, but is coded by pointers P% to save moving three array elements (node%, count%, dad%) at each exchange.

The output is to another named file, and also shown on the screen (lines 5060 and 5010) but is rather slow because bit-handling is very slow in Basic.

Next issue we will have a look at a decoding routine for this program as we extend the data sent to the output file to include the tree shape.

```
   10 REM HUFFMAN ENCODER
   20 REM Version B1.0
   30 REM Author  Bernard Hill
   40 REM Beebug  March 1991
   50 REM Program subject to copyright
   60 :
  100 DIM code%(255),len%(255),tot%(255)
  110 DIM count%(511),node%(511),P%(511),dad%(511)
  120 INPUT "Input file name: "n$
  130 f=OPENINn$:IF f=0 THEN PRINT"File not found":END
  140 PRINT"Analysing file..."
  150 REPEAT x=BGET#f:tot%(x)=tot%(x)+1
  160 UNTIL EOF#f:Len%=EXT#f:CLOSE#f
  170 PRINT"Building tree..."
  180 FOR k=0 TO 511:node%(k)=-1:NEXT
  190 k=0:FOR x=0 TO 255
  200 IF tot%(x)>0 THEN k=k+1:node%(k)=x:count%(k)=tot%(x)
  210 NEXT:FOR x=1 TO k:P%(x)=x:NEXT
  220 start=1:REPEAT
  230 PROCsort(start,k):k=k+1:P%(k)=k
  240 count%(k)=count%(P%(start))+count%(P%(start+1))
  250 dad%(P%(start))=-k:dad%(P%(start+1))=k
  260 start=start+2:UNTIL start=k
  270 PROCfilltree
  280 PROCxlate
  290 END
  300 :
 1000 DEF PROCsort(begin%,end%)
 1010 LOCAL j%,x%,done%
 1020 REPEAT done%=TRUE:begin%=begin%+1
 1030 FOR j%=end% TO begin% STEP -1
 1040 IF count%(P%(j%-1))>count%(P%(j%)) THEN x%=P%(j%):P%(j%)=P%(j%-1):P%(j%-1)=x%:done%=FALSE
 1050 NEXT:UNTIL done% OR begin%=end%
 1060 ENDPROC
```

Continued on page 52

BEEBUG Function/Procedure Library

by Stefano Spina

INTRODUCTION

This month we introduce a new regular feature to BEEBUG magazine in the form of a library of functions and procedures. In this and the immediately following issues we will be presenting routines contributed by Stefano Spina, who has provided the impetus and many routines to get us started.

However, we welcome the submission of further collections of function and procedure definitions, and all those which we publish will be duly acknowledged and the authors paid at our normal rates. We must insist that all contributions for this library consist of fully documented routines corresponding as closely as possible to the style and layout of the examples presented this month. A library such as this works much better if all routines (both program code and documentation) conform to a common standard.

We would also welcome routines built around a common theme, and there is also no reason why the routines in such a collection should not be inter-dependent. Just make this clear in the documentation (as in some of the examples here).

In presenting the routines we have attempted to adopt a fairly readable style. As a consequence many spaces (and some other characters) have been included where this makes the routines easier to follow. If memory is at a premium you will find many opportunities to save space, though the simplest and most thorough way to achieve this is via the *Squeeze* command in BEEBUG's EdiKit (or similar).

In organising your version of the BEEBUG Function/Procedure library, you may find it best to save each routine as a separate file using the name of the routine as the file name. You will also have to choose whether to save as tokenised Basic, or spooled out in ASCII format. The choice is yours, but on balance we might suggest the latter. However, you are quite free to organise the library as you wish. Note that because of the limited 31 file catalogue permitted by the DFS, all the routines from each issue will be included on the accompanying magazine disc as a single Basic program.

A library of functions and procedures provides an easy way of developing your own programs, even if you find you need to make some modifications. And even if you do not use them directly you may still find many useful ideas for your own programs among the routines listed here.

THE FUNCTION/PROCEDURE LIBRARY

This month we start with a number of routines which are primarily (though not exclusively) concerned with the formatting and manipulation of character strings.

Routine 1: Big1
Type: PROCEDURE
Syntax: PROCbig1(S$,C%,R%,H%)
Purpose: **Prints double-height strings in modes 7/135**
Parameters: S$ String to be printed
C% Column
R% Row
H% Teletext code for text colour (129-135)
Notes: None
Related: None

```
10 MODE 135
20 str$="Program Title"
30 PROCbig1(str$,5,12,131):
REM print yellow string at col. 5, row 12
```

* * *

Routine 2: Big2
Type: PROCEDURE
Syntax: PROCbig2(S$,C%,R%)
Purpose: **Prints double-height strings in any mode except for modes 7/135**
Parameters: S$ String to be printed
C% Column
R% Row

Notes:	None	
Related:	None	

```
10 MODE 129
20 str$="Program Title"
30 PROCbig2(str$,5,12):
REM string printed at col. 5, row 12
```

* * *

Routine 3:	**Box**	
Type:	PROCEDURE	
Syntax:	PROCbox(X1%,Y1%,X2%,Y2%)	
Purpose:	**Draws vertical/horizontal lines or boxes in graphics modes**	
Parameters:	X1%	Left horiz. co-ordinate
	Y1%	Left vert. co-ordinate
	X2%	Right horiz. co-ordinate
	Y2%	Right vert. co-ordinate
Notes:	Co-ordinates must be graphic (0-1279 horiz. 0-1023 vert.).	
Related:	None	

```
10 MODE 128
30 PROCbox(100,100,800,800):
   REM draws a box
40 PROCbox(100,600,800,600):
   REM draws a horizontal line at y=600
50 PROCbox(500,100,500,800):
   REM draws a vertical line at x=500
```

* * *

Routine 4:	**Centre**	
Type:	FUNCTION	
Syntax:	FNcnt(S$,W%)	
Purpose:	**Centres a string on the screen or within a given width.**	
Parameters:	S$	String to be centred.
	W%	Width within which string must be centred. If W% is set to zero then the complete screen width is used.
Notes:	None	
Related:	FNmd	

```
10 MODE 128
20 S$="Centre This"
30 PRINT FNcnt(S$,40):
   REM string centred into the given width
   (40 col.)
40 PRINT FNcnt(S$,0):
   REM string centred into the whole
   screen width
```

* * *

Routine 5:	**Colour**	
Type:	PROCEDURE	
Syntax:	PROCcolour(M%)	
Purpose:	**Reverses background/foreground from black/white to white/black**	
Parameters:	M%	Flag for reverse/normal 0 Normal output 1 Reverse output
Notes:	Can be used in any mode except for mode 7/135 and in any case the colour swap is between black and white.	
Related:	FNmd	

```
10 MODE 128
20 S$="Reverse This"
30 PROCcolour(1):PRINT S$:
   REM string printed in reverse
   (white/black)
40 PROCcolour(0):PRINT S$:
   REM string printed in normal
   (black/white)
```

* * *

Routine 6:	**Curs**	
Type:	PROCEDURE	
Syntax:	PROCcurs(M%)	
Purpose:	**Makes the cursor visible or invisible**	
Parameters:	M%	Flag to set reset cursor's status: 0 The cursor is invisible 1 The cursor is visible
Notes:	None	
Related:	None	

* * *

Routine 7:	**Cursor**	
Type:	PROCEDURE	
Syntax:	PROCcursor(M%)	
Purpose:	**Cursor's operation control**	
Parameters:	M%	Flag to set operation 0 Normal operation 1 Horizontal reverse output 2 Vertical reverse output 3 Horizontal/vertical reverse output
Notes:	None	
Related:	None	

* * *

Routine 8: **Flash**
Type: PROCEDURE
Syntax: PROCflash(S$,C%,R%)
Purpose: **Displays flashing string in any mode except for mode 7/135**
Parameters: S$ String to be flashed
C% Column
R% Row
Notes: The required string flashes at the required column and row until a cursor key or Return is pressed. The "get%" global variable contains the ASCII code of the pressed key. The required keys can be changed to perform other actions on exit, depending upon the user's requirements.
Related: PROCcolour, PROCcurs, FNtest

```
 90 S$="Flashing string"
100 PROCflash(S$,10,10)
110 IF get%=13 THEN PROCreturn:
    REM Return key press
120 ON get%-135 PROCleft,PROCright,
    PROCdown,PROCup:
    REM Cursor key press
```

* * *

Routine 9: **Fonts**
Type: PROCEDURE
Syntax: PROCfont(S$,C%,R%,M%)
Purpose: **Displays a string to be printed underlined, in bold, or both in any graphics mode.**
Parameters: S$ String to be printed
C% Column
R% Row
M% Flag to set required output
0 Underlined string
1 Emboldened string
2 Both attributes
Notes: Allowed modes are 0, 1, 2, 4, 5
Related: FNtest

```
 90 S$="Output string"
100 PROCbold(S$,10,10,0):
    REM underlined string
110 PROCbold(S$,10,12,1):
    REM bold string
120 PROCbold(S$,10,14,2):
    REM both underlined/bold string
```

* * *

Routine 10: **Mode**
Type: FUNCTION
Syntax: FNmd
Purpose: **Returns the actual screen mode (1/7)**
Parameters: None
Notes: The result is always between 1 and 7 irrespective of whether normal/shadow memory is used.
Related: None

```
10 MODE 0
20 PRINT FNmd:REM the result is 0
20 MODE 129
30 PRINT FNmd:REM the result is 1
```

* * *

Routine 11: **Test**
Type: FUNCTION
Syntax: FNtest(C%,G%)
Purpose: **Returns a value of TRUE if the ASCII code of key pressed matches the required value irrespective of its case, else returns FALSE.**
Parameters: C% Test ASCII value
G% Given ASCII key value
Notes: Is useful when a certain key must be pressed to perform a certain action.
Related: None

```
100 REPEAT
110 get%=GET
120 UNTIL FNtest(65,get%):
    REM exits only if keypress is "A" or
"a"
```

* * *

```
20000 :
20010 REM Big1
20020 :
20030 DEF PROCbig1(S$,C%,R%,H%)
20040 S$=CHR$141+CHR$H%+S$
20050 PRINTTAB(C%,R%)S$
20060 PRINTTAB(C%,R%+1)S$
20070 ENDPROC
20080 :
20090 REM Big2
20100 :
20110 DEF PROCbig2(S$,C%,R%)
20120 LOCAL A%,B%,I%,X%,Y%
20130 VDU31,C%,R%:B%=&8F
```

```
20140 FOR I%=1 TO LEN(S$)
20150  ?B%=ASC(MID$(S$,I%,1))
20160  A%=10:X%=B%:Y%=B% DIV &100
20170  CALL&FFF1
20180  VDU23,141,B%?1,B%?1,B%?2,B%?2,B%?
3,B%?3,B%?4,B%?4:VDU141,10,8
20190  VDU23,141,B%?5,B%?5,B%?6,B%?6,B%?
7,B%?7,B%?8,B%?8:VDU141,11
20200 NEXT I%
20210 ENDPROC
20220 :
20230 REM Box
20240 :
20250 DEF PROCbox(A%,B%,C%,D%)
20260 MOVE A%,B%:DRAW A%+D%,B%
20270 DRAW A%+D%,B%+C%:DRAW A%,B%+C%
20280 DRAW A%,B%
20290 ENDPROC
20300 :
20310 REM Centre
20320 :
20330 DEF FNcnt(S$,W%):LOCAL S%
20340 IF W%=0 S%=FNmd ELSE =INT(W%-LENS$
)/2
20350 IF S%=0 OR S%=3 W%=80 ELSE IF S%=2
O RS%=5 W%=20 ELSE W%=40
20360 =INT(W%-LENS$)/2
20370 :
20380 REM Colour
20390 :
20400 DEF PROCcolour(M%)
20410 LOCAL B%,F%,S%
20420 S%=FNmd:IF S%=7 ENDPROC
20430 IF S%=0 OR S%=3 OR S%=4 OR S%=6 B%
=129:F%=1 ELSE IF S%=1 OR S%=5 B%=131:F%
=3 ELSE B%=135:F%=7
20440 IF M%=0 COLOUR F%:COLOUR 128 ELSE
COLOUR0:COLOUR B%
20450 ENDPROC
20460 :
20470 REM Curs
20480 :
20490 DEF PROCcurs(M%)
20500 VDU23,1,M%;0;0;0;0;
20510 ENDPROC
20520 :
20530 REM Cursor
20540 :
20550 DEF PROCcursor(M%)
20560 VDU23,16,M%;0;0;0;0;
20570 ENDPROC
20580 :
20590 REM Flash
20600 :
20610 DEF PROCflash(S$,C%,R%)
20620 LOCAL F%:F%=TRUE:PROCcurs(0):*FX4,
1
20630 REPEAT
20640 PROCcolour(1)
20650 PRINTTAB(C%,R%)S$:get%=INKEY(40)
20660 IF get%=13 OR get%>=136 AND get%<=
139 OR FNtest(77,get%) OR FNtest(69,get%
) F%=FALSE
20670 PROCcolour(0)
20680 PRINTTAB(C%,R%)S$:get%=INKEY(40)
20690 IF get%=13 OR get%>=136 AND get%<=
139 OR FNtest(77,get%) OR FNtest(69,get%
) F%=FALSE
20700 UNTIL NOT F%
20710 PROCcurs(1):*FX4,0
20720 ENDPROC
20730 :
20740 REM Fonts
20750 :
20760 DEF PROCfont(S$,C%,R%,M%)
20770 LOCAL B%,D%,S%:S%=FNmd
20780 IF S%=3 OR S%=6 OR S%=7 ENDPROC
20790 IF S%=0 B%=1:D%=2 ELSE IF S%=1 OR
S%=4 B%=2:D%=4 ELSE B%=4:D%=8
20800 C%=16*B%*C%:R%=(32*(32-R%))-4
20810 VDU5:MOVE C%,R%:PRINT S$
20820 IF (M%=0 OR M%=2) THEN PROCfnt1 EL
SE PROCfnt2
20830 ENDPROC
20840 :
20850 DEF PROCfnt1
20860 MOVE C%,R%:PRINT STRING$(LEN(S$),"
_")
20870 IF M%=0 VDU4
20880 ENDPROC
20890 :
20900 DEF PROCfnt2
20910 MOVE C%+D%,R%:PRINT S$:VDU4
20920 ENDPROC
20930 :
20940 REM Mode
20950 :
20960 DEF FNmd
20970 LOCAL A%,Z%:A%=&87:Z%=USR(&FFF4)
20980 =VAL(LEFT$(STR$~(Z% AND &00FF0000)
,1))
20990 :
21000 REM Test
21010 :
21020 DEF FNtest(C%,G%)
21030 IF (G% AND 95)=C% =TRUE ELSE =FALS
E
21040 :
```

512 Forum

by Robin Burton

I've decided this month to start dealing with a few topics about which I have had several queries recently and hopefully they'll be of interest to many of you. Before that though an explanation is called for. To be able to deal with these items I have come to the conclusion that for 1991 a slight shift of policy is called for concerning the potential content of the Forum.

RULES OF PLAY

In the past, the range of topics covered in the Forum has on occasion been restricted. This is a result of the fact that I have, in general, avoided mentioning other 512 activities or projects in which I am or have been involved.

From this month onwards I have decided to ignore this factor as and when it's justified, specifically when it stops me writing about items that should be included in the Forum. You'll all know that I wrote the Dabs Press 512 Technical Guide and you will probably also know of my Essential Software activities. It seems unreasonable (and to one or two of you as well, judging by your letters) to treat anything with any connection to either of these as taboo.

After all, I write 512 Forum which is the only magazine column which has ever been devoted to the 512, while Dabs Press is the only book publisher to have catered for 512 users and Essential Software is the only supplier with a wide range of products exclusively for the 512. It seems to me that to pretend in 512 Forum that two out of these three don't exist is ridiculous.

The Forum will not become an advertisement, that I guarantee, but the fact is that I get genuine queries from BEEBUG members who refer, directly or otherwise, to various products with which I have some involvement. Some of these queries, like this month's, are concerned with aspects of using and even worse, selling (!) a 512 which potentially are of general interest to a fairly wide audience.

Until now I've avoided such items in the Forum, but the change of approach is because this is becoming more and more unrealistic as time goes by. However, I will also add that if you don't think that this is a legitimate attitude read this month's Forum before you finally decide. If afterwards you still think it's not a reasonable change of rules, write to me or to the Editor and say so. This is after all a Forum where any 512 user can air his or her views.

It's also worth repeating that if there's any particular topic or item that you'd like me to cover in the Forum, let me know and I'll do my best to include it. I don't claim to know everything about the 512, particularly so at the level of specific applications, or even in some general areas like DTP, but if I don't know the answers I can always appeal to members for information.

UPGRADING?

This is the first item I'll comment about which indirectly has something to do with me, because it seems some of you have been having traitorous thoughts. I've had several queries over the last few months about the advisability (or otherwise) of moving to an Archimedes or an A3000 and using the PC emulator instead of a 512 to run DOS software. I'd guess that this idea has crossed the mind of quite a large number of you from time to time.

Three very recent queries from members spring to mind because their thoughts, questions and the possible alternatives they were considering were very similar.

Basically the question was, "I'm considering expanding the memory of the 512 and perhaps adding a hard disc too, but would I be better off instead selling my BBC micro and 512, replacing both of them with an A3000 and using the PC emulator for DOS?".

Of course I'm biased and so the instant answer is "Certainly not!", but joking aside this is, or will be sooner or later, a question for most of us. It's the most logical and obvious progression from the 8-bit BBC micro and the 512 to a single machine.

It's not for me to make this sort of decision for anyone else, and bear in mind my admitted bias, but I do think there are a number of points which are not very well publicised but which you should clearly understand before you begin to weigh up the pros and cons of such a change. Naturally we're concerned with DOS in this column, so that's what we'll concentrate on.

THE EMULATOR

First, forget all other considerations. Let's just look at one or two important facts about the Archimedes and the DOS emulator that you might not have realised.

We all know that the ARM processor is a very high speed device and that the Archimedes or the A3000 (even limiting ourselves to the ARM2 processor) can outpace many more expensive machines in numerous benchmark tests. However, programs for such tests are written in each machine's own language for obvious reasons, so first of all don't be misled into thinking that these test results give a direct indication of how fast the PC emulator runs DOS software, they don't. To understand why this is so it helps to know what a software emulator does.

Obviously the ARM processor can't directly execute (for example) 80186 machine code instructions; they are meaningless to an ARM processor because the two chips are entirely different both internally and in how they address and access memory. Also, the 80186 and the later '286, '386 and '486 variants are what are known as CISC processors (Complex Instruction-Set Computer) which means that a single machine code instruction can effectively perform several quite complicated operations simultaneously, such as, for example:

"Repeatedly compare the contents of indexed memory location A with the contents of indexed memory location B. Do this for up to 65,535 repeats incrementing both counters as you do so, or until the contents of locations A and B are equal (or not equal, if you prefer)"

If you're not familiar with '86 series machine code let me assure you that all the above operations really can be carried out by a single instruction by your 512.

What the emulator must do (remember that the emulator is a program itself) is to read the 80186 processor instructions from the program, getting also any registers and/or memory locations used, then to translate these into simpler more elementary operations which will be understood by the RISC (Reduced Instruction-Set Computer) processor of the Archimedes. These translated operations can then finally be executed in ARM code.

The result of this sort of technique, given the very different nature of the two processors, is that a single '86 series instruction could quite possibly generate a dozen ARM instructions, involving quite a lot of memory storage and retrieval of intermediate data along the way. You must also add the overhead of reading each ARM instruction involved in the process, possibly a large number of times, before execution of the 'original' single instruction is complete. On top of that the emulator must similarly run a version of DOS by the same sort of laborious process. The easiest way to think of this process in 'BBC terms' is as interpreted assembly code and you know how fast BBC machine code executes compared with interpreted Basic.

The result is that when running under the emulator, even allowing for the impressive speed of the ARM processor, an A3000 or an Archimedes will be about four or five times slower than your 512 for most operations. I believe the current version of the emulator runs at an effective speed of around 2MHz or perhaps a bit less, while the 512 runs at 10MHz, so simple arithmetic will tell you what sort of comparative performance you can expect.

SCREEN DISPLAY

Next let's consider screen output. Again we all know that the ARM family of Acorn micros are capable of some quite stunning graphics displays with large numbers of colours on screen at the same time, the equivalent in fact of a PC VGA display. Quite so, but it's not so in the PC emulator. What you get in fact, is exactly the same display capability as you have in your 512. In other words text can only be shown in a single colour and graphics can use up to four colours. The emulator provides only CGA support, which is exactly what you get from your 512 and BBC micro, so there's nothing to gain in that area either.

A further point is that, as I understand it, you don't get as much free (DOS) RAM in the Archimedes under the emulator as you can in an expanded 512. You need extra memory in an A3000 to end up with as much DOS memory as a one megabyte 512, so don't forget to add that cost into your calculations.

DISCS

The other item in this comparison was of course a hard disc, so let's make sure we compare like with like. For an A3000 to match the disc speed and facilities offered by a 512 when fitted with a hard disc you obviously would need a hard disc, plus possibly a second floppy drive for the 3000 as well.

This isn't technically any problem of course, but have you seen the prices of Archimedes hard discs? - something else which must be considered very carefully. It should be said that you might be in the position of having to buy a hard disc for your 512 to make this a fair comparison, but given that plenty of other BBC micro users are moving to the Archimedes on a regular basis I'd think a second-hand hard disc for the BBC might be possible to find.

I don't want anyone to think I'm criticising the Archimedes, I'm not, that would be inaccurate. However, the majority of 512 users probably don't use their machine in business, so the comparison ultimately is one of cost and what you get for your money. I was asked whether adding a memory expansion (£99.00) and a hard disc (say £250.00 second-hand) to the 512 was a serious alternative to the 'obvious' progression to an Archimedes with the DOS emulator.

DECISION

The cost of the 512 option comes to well under £400.00 complete, because you've already got most of it. Now let's consider the cost of the A3000 system. Let's assume that you can continue to use your existing monitor (if not add that to the bill) and let's assume you restrict yourself to an A3000, with a hard disc, a second floppy (if needed to match your 512), the emulator itself plus a supply of 3.5 inch discs. You might well spend up to £1,200.00 (though even an A410 with 2MBytes of memory and a hard disc will only cost some £1360 with a PC emulator) - and you can expect a trade-in value of perhaps £300 or more on your existing system.

Now let's weigh-up the benefits of the move. You'll get a system that performs (in DOS) roughly five times slower, the display facilities will be the same and you'll be considerably poorer. It is true that the emulator provides for a later version of DOS, but expanding the 512 seems to overcome quite a few compatibility problems anyway. Based only on using the system in DOS I don't think there's any decision to make.

Of course I haven't considered any pure 'BBC' aspects in this discussion, they vary from user to user so you'll have to make your own decisions about them. It is true that a good deal of your existing BBC software can be run on an Archimedes, but those programs that do run offer no more memory than when you use them in your current system.

Even so, don't automatically think all your software will run. For a simple example look no further than Inter-Word, where you'll need a special (i.e. new) version. In addition, some programs won't run and have no equivalent Archimedes version, so you might also need to buy new software to regain what you already had, but at yet more cost.

Of course, if what you are really after is a dedicated DOS environment, then you might be better going for a genuine PC compatible anyway (Editor).

A Treasurer's Companion (continued from page 14)

Listing 2

```
3000 DEF FNN="DAT1"
3010 DIM A$(10)
3020 J%=0:READ A$(10):REPEAT:J%=J%+1:FO
R I%=1 TO 10:READ A$(I%)
3030 NEXT:PRINT A$(1):UNTIL A$(10)="100
":PRINT J%
3040 DATA 1,Miss Belvaney
3050 DATA 16 Heavitree Road,G,1KT 12SJ,
,,10/1/90,2,A
3060 DATA 1,Miss Bravassa
3070 DATA 2 Oakley House,Densworth Road
,S,5KT 10AB,,26/9/89,2,A
3080 DATA 1,Miss Madeline Bray
3090 DATA 25 St Martins Road,G,2KT 23JK
,,,26/9/89,2,A
3100 DATA 1,John Browdie
3110 DATA 5 Glenhurst Mansions,Dallas R
oad,G,3KT 2AB,,25/9/90,2,A
3120 DATA 1,Mr Frank Cheeryble
3130 DATA 5 Rothwell Street,G,5KT 10JZ,
,,10/9/89,2,A
3140 DATA 1,Mr Crowl
3150 DATA 15 St Martins Road,G,7KT 8VS,
,,,,A
3160 DATA 2,Mr and Mrs Crummles
3170 DATA Alma House,Alma Square,S,10KT
7XY,211 3165,,,L
3180 DATA 1,Mr Folair
3190 DATA 13 Kelvon House, Blenheim Cre
scent,S,9KT 3AD,,3/10/89,2,A
3200 DATA 1,Mr Gregsbury
3210 DATA 15 Tideswell Road,G,7KT 7BC,,
,29/9/89,2,A
3220 DATA 1,Arthur Gride
3230 DATA Flat 5,2 Millet Avenue,G,3KT
7SV,,,,A
3240 DATA 1,Mrs Grudden
3250 DATA 2 Kirkham Avenue,S,8KT 7DF,,,
4/12/89,2,A
3260 DATA 1,Sir Mulberry Hawk
3270 DATA The Grange, Mulberry Road,S,7
KT 2SE,201 3726,,,A
3280 DATA 2,Mr and Mrs Kenwig
3290 DATA 119 Court Lane,G,1KT 5SJ,,,4/
12/89,4,A
3295 DATA 100
```

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

We have a number of mostly minor points this month relating to recently published programs.

PHONE CALL COSTING (VOL.9 NO.6)

The version of this program for the model B on the magazine disc inadvertently still contained a line which is only relevant to the Master version. To cure this just delete line 700.

One reader wished to use phone numbers longer than could be accommodated - to do this amend the value as appropriate in line 980, and amend lines 3540 to 3580 to produce a wider box.

MIKROTEL (VOL.9 NOS.7 & 8)

Line 1080 of the *Compact* program should be amended to read:

```
1080 *SAVE MCCMPCT
```

As listed, the main program is otherwise unable to find this assembled code.

In the short program to create the HASH file (Vol.9 No.7 p.17), line 70 should read:

```
70 FOR I%=0 TO &7000
```

Finally, the following function definition was omitted by the author and should be appended to your completed Mikrotel program:

```
4640 DEFFNrecno(A%)
4650 LOCALI%
4660 REPEAT
4670 I%=I%+1
4680 UNTIL(!(FNh(I%)+14)AND&FFFF)=
     A%ORI%=NR%+1
4690 =I%
```

HINTS & TIPS (VOL.9 NO.8)

Under 'Quick and Easy File Save' line 30 should contain '?N=13' and not '?N=130'

POSTBAG (VOL.9 NO.8)

Editing Mr.Robertson's letter unfortunately removed some essential quotes. The relevant sections should read ||!LM7|!" and ||" and not as printed.

Practical Assembler (Part 9): More Filing System Commands

by Bernard Hill

Last month we introduced a routine which made a second copy of a file under a new name with the format:

```
*FCOPY file1 file2
```

It was very slow for large files because it copied a file byte by byte. This month we introduce an alternative routine called by a new command, *FC, which performs the same task but *LOADs the file and then *SAVEs it under the new name, carefully preserving execution and load addresses.

In MOS terms, copying FILE1 to FILE2 means doing as the following example:

```
PRINT ~PAGE
1900
    ( we load the file here )

PRINT ~HIMEM
7C00

PRINT ~HIMEM-PAGE
6300
    (the largest file we can copy is
    thus &6300 bytes long)

*INFO FILE1
$.FILE        000E00 000E2B 001440 058
    (load address &E00, execution &E2B,
    length &1440 - well under our &6300
    bytes)

*LOAD FILE1 1900
*SAVE FILE2 1900 2D40 E2B E00

    (the end address is 1900+1440=2D40)
```

Let's look in turn at these operations when implemented in assembler.

AVAILABLE MEMORY

PAGE is a Basic keyword (pseudo-variable); the equivalent in machine code is the splendid term OSHWM (OS High Water Mark!) which represents the lowest available user memory, taking into account any font explosions or private and public RAM workspaces. It is read with Osbyte 131, the relevant value being returned in X and Y:

```
A%=131:PRINT ~USR(&FFF4)
 B1190083
        (i.e. 1900)
```

Similarly the value of HIMEM is obtained using Osbyte 132:

```
A%=132:PRINT ~USR(&FFF4)
 B17C0084
```

and the difference gives the available space:

```
LDA #&83 : JSR &FFF4 \ find oshwm
STX oshwm : STY oshwm+1
LDA #&84 : JSR &FFF4 \ find HIMEM
SEC : TXA : SBC oshwm : STA space
TYA : SBC oshwm+1 : STA space+1
```

Now having a measure of the largest available memory, we need to find out about our file (*INFO). The key to this is the system Osfile routine at &FFDD mentioned last month which has a number of functions depending on the value of A on entry (the X and Y registers point to a parameter block - 'PB'). The full list of functions is:

A=0	do *SAVE
A=1	write PB to disc catalogue
A=2	write Load Address to disc catalogue
A=3	write Exec Address to disc catalogue
A=4	write file attributes to catalogue
A=5	read disc catalogue (i.e. do *INFO)
A=6	delete file
A=7	create empty file (Master only)
A=&FF	do *LOAD

The parameter block is:

Bytes 0 - 1	:address of the characters of the filename, ending in &0D
Bytes 2 - 5	:load address of the file
Bytes 6 - 9	:execution address of the file
Bytes 10-13	:length of file or start address if saving
Bytes 14-17	:end address of file if saving.

Having obtained the file name starting at address "filein" we can do a *INFO by:

```
LDA #filein MOD 256 : STA pblock
LDA #filein DIV 256 : STA pblock+1
LDX #pblock MOD 256
LDY #pblock DIV 256
LDA #5 : JSR &FFDD
```

and the parameter block will then contain the load and execution addresses and the file length. Now we must compare the file length with the available space to make sure it fits. If it does not, our program will give a "file too big" error and stop.

Now if all is well we can load the file, but we shall need to alter the load address to OSHWM, and so we use a second parameter block since we shall need the information in the first to be preserved for a while. Here is a comparison between the two parameter blocks (unused or irrelevant fields are left blank):

	pblock	pblock2
0-1	filename1	filename1
2-5	file's load addr	OSHWM
6-9	file's exec addr	0
10-13	file's length	
14-17		

(Note: if the 6th byte of the parameter block is zero on a LOAD, then a load to the specified address - e.g. *LOAD FILE1 1900 - will be performed, otherwise the file will load to its own address as in *LOAD FILE1)

This requires some simple byte copying before calling Osfile with A=&FF.

For *SAVE we will use the first parameter block again, but we need to change the contents of this slightly as the format of a *SAVE (A=0) is not quite the same as for *INFO (A=5):

	Old pblock (after *INFO)	New pblock (for *SAVE)
0-1	filename1	filename2
2-5	file's load addr	
6-9	file's exec addr	file's exec addr
10-13	file's length	data start addr
14-17		data end addr

The data start address is of course OSHWM, and the end address must be calculated from the file length added to OSHWM. Finally we call Osfile with A=0 to *SAVE the file.

Listing 1 shows this program in the by now well-known sideways ROM format. When an unknown star command is issued to the MOS, the MOS issues a JSR &8003 to all the service ROMs in turn with A set to 4 and (&F2),Y pointing to the command. This ROM responds to the command "*FC" and traps a bare '*FC' command to give a helpful message, but otherwise expects the command to be followed with two file names as in last month's article. The checking of the characters F and C is performed in lines 270-290 and if successful, control continues to the filename checking routines identical to those of last month's disc-based system (lines 300-560).

The other functions mentioned above are then performed in a very obvious manner (lines 570-1070). The only difference is in the error handling, when in sideways RAM, as mentioned in part 2 of this series - see Vol.9 No.2. As there are a number of error returns, I have used the excellent sideways RAM error subroutine sent in by Andrew Rowland in the Hints and Tips section in Vol.9 No.4 on p.61. In case the image is ever frozen into ROM we omit variable allocation within the assembler routine (as in ".oshwm EQUW 0"), but use some workspace from the beginning of page 9 (lines 110-160).

EXTENSIONS TO THE PROGRAM

I leave these to you as a fairly simple exercise. They might include:

1. A sideways ROM help response for the ROM to announce itself on *HELP. The entry point is when A=9. Remember to exit with A intact so that lower ROMs can also respond to *HELP.

2. A non-fatal operation if the file won't fit into available memory. You could use the OSBGET-OSBPUT routines from last month's article, or if feeling more adventurous get out your *Advanced User Guide* and implement a system of loading part of the file at a time with OSGBPB, but be warned, it's only slightly faster than the BGET-BPUT method on a standard DFS. as it still loads a byte at a time but minimising disc head movement.

```
   10 REM File copier (SWR version)
   20 REM Version B2.1
   30 REM Author  Bernard Hill
   40 REM BEEBUG  March 1991
   50 REM Program subject to copyright
   60 :
  100 MODE7
  110 oshwm  =&900:REM 2 bytes
  120 space  =&902:REM 4 bytes
  130 pblock =&906:REM 18 bytes
  140 pblock2=&918:REM 18 bytes
  150 filein =&92A:REM 20 bytes
  160 fileout=&93E:REM 20 bytes
  170 FOR opt=4 TO 7 STEP 3
  180 P%=&8000:O%=&7000:Q%=O%
  190 [OPT opt
  200 BRK:BRK:BRK:JMP start
  210 EQUB &82:EQUB copyright :EQUB 0
  220 EQUS "File Copy ROM":EQUB 0
  230 .copyright EQUB 0
  240 EQUS "(C) BEEBUG March 1991":BRK
  250 .start CMP #4:BEQ command:RTS
  260 .command PHA:TXA:PHA:TYA:PHA
  270 LDA (&F2),Y:AND #&DF:CMP #ASC"F"
  280 BNE exit:INY
  290 LDA (&F2),Y:AND #&DF:CMP #ASC"C"
  300 BNE exit:INY
  310 LDA (&F2),Y:CMP #32:BEQ go
  320 CMP #13:BNE exit
  330 LDX #0:.loop LDA info,X:BEQ finis
  340 JSR &FFE3 : INX : JMP loop
  350 .info EQUS "Syntax: *FC <infile> <
outfile>":EQUW 13
  360 .exit PLA:TAY:PLA:TAX:PLA:RTS
  370 .finis PLA:TAY:PLA:TAX:PLA:LDA #0:
RTS
  380 .go DEY \ first find file names
  390 .skipsp INY:LDA (&F2),Y
  400 CMP #32:BEQ skipsp:LDX #0
  410 .file1 STA filein,X
  420 INX:INY:LDA (&F2),Y
  430 CMP #13:BEQ nofile2
  440 CMP #32:BEQ endfile1:BNE file1
  450 .nofile2 JSR error:EQUB 67
  460 EQUS "Filename missing":EQUB 0
  470 \ place &0D on end of name
  480 .endfile1 LDA #13:STA filein,X
  490 \ second file
  500 .loop INY:LDA (&F2),Y
  510 CMP #32:BEQ loop:LDX #0
  520 .file2 STA fileout,X
  530 INX:INY:LDA (&F2),Y
  540 CMP #13:BEQ endfile2
  550 CMP #32:BNE file2
  560 .endfile2 LDA #13:STA fileout,X
  570 \ evaluate how much space we have
  580 \ read OSHWM ("PAGE")
  590 LDA #&83:JSR &FFF4 \ osbyte
  600 STX oshwm:STY oshwm+1
  610 \ read HIMEM
  620 LDA #&84:JSR &FFF4 \ osbyte
  630 TXA:SEC:SBC oshwm:STA space
  640 TYA:SBC oshwm+1:STA space+1
  650 LDA #0:STA space+2:STA space+3
  660 \ how big is file?
  670 LDA #filein MOD 256:STA pblock
  680 LDA #filein DIV 256:STA pblock+1
  690 LDX #pblock MOD 256
  700 LDY #pblock DIV 256
  710 LDA #5 \ read catalog info on file
  720 JSR &FFDD \ osfile
  730 CMP #0 \ does file exist?
  740 BNE ok:JSR error:EQUB 214
  750 EQUS "File not found":EQUB 0
  760 .ok \ length in pblock+10 to 13
  770 \ do dummy subtraction:
  780 LDX #0: LDY #3: SEC \ 4 byte nos
  790 .loop LDA space,X:SBC pblock+10,X
  800 INX : DEY : BNE loop
  810 BCS ok2 : JSR error: EQUB 255
  820 EQUS "File too big":EQUB 0
  830 .ok2 \ now use pblock2 for load
  840 LDA #0:LDX #17:.loop
  850 STA pblock2,X:DEX:BPL loop
  860 LDA pblock : STA pblock2
  870 LDA pblock+1 : STA pblock2+1
  880 LDA oshwm:STA pblock2+2
  890 LDA oshwm+1:STA pblock2+3
  900 LDX #pblock2 MOD 256
  910 LDY #pblock2 DIV 256
  920 LDA #&FF \ osfile HIMEMinstructn
  930 JSR &FFDD
  940 \now *ENDIF file OSHWM end exec ol
d
  950 \change 1st pblock, calc end addr
  960 CLC:LDA oshwm:ADC pblock+10
  970 STA pblock+14:LDA oshwm+1
  980 ADC pblock+11:STA pblock+15
  990 LDA oshwm: STA pblock+10
 1000 LDA oshwm+1:STA pblock+11
 1010 LDA #0:STA pblock+16:STA pblock+17
 1020 STA pblock+12:STA pblock+13
 1030 LDA #fileout MOD 256:STA pblock
 1040 LDA #fileout DIV 256:STA pblock+1
 1050 LDX #pblock MOD 256
 1060 LDY #pblock DIV 256
 1070 LDA #0:JSR &FFDD \ osfile save
 1080 JMP finis
 1090 .error PLA:STA &A8:PLA:STA &A9
 1100 LDY #0:STY &100
 1110 .loop INY:LDA (&A8),Y:STA &100,Y
 1120 BNE loop:JMP &100
 1130 ]:NEXT
 1140 c$="SAVE FCROM "+STR$~Q%+" "+STR$~
O%+" 8000 8000"
 1150 PRINTc$:OSCLIc$
```

BEEBUG Education

Mark Sealey takes a look at Prestel and the benefits of Campus 2000.

PRESTEL EDUCATION

Many teachers at present are in danger of suffering from A4-itis... undue bombardment from all quarters by masses of information, catalogues and news of innovations. So this month BEEBUG Education presents its topic - where to get yet more such information - very warily.

This column looked (Vol.9 no.1) at the NERIS on-line information service, which is accessed direct and is concerned chiefly with curriculum material. During recent months, the price of modems has fallen and it has become more usual for many schools and colleges to decide to use a portion of their IT budget on several such services.

One of these, which deserves your attention if you do not already know of it, is provided by Campus 2000 on Prestel. Hence its format is viewdata (teletext frame-based, rather than scrolling) but is none the worse for that. Videotex has come a long way since the early days of the BBC Teletext service and a collection of rather unappealing pages in a hard to find corner of Micronet on Prestel.

Prestel Education begins on page 888 and can either be obtained by typing *888# (hash) or *EDUCATION#. From that header page two apparent routes are possible: Campus 2000 itself and a subject index to education services on Prestel. In fact keying the option for the latter also leads you to the pages provided under the Campus umbrella.

And a very wide umbrella it is indeed. At this point there are several guides, quick guides and A-Z indices as well as 'What's New' and 'Information' type pages.

There are some two dozen major areas of interest and sources of information. Indices on pages 88800 and 88862 give some idea of what information is available and which organisations have pages on the main database. These range from the familiar - an on line UCCA (Universities Central Council on Admissions) database (actually on page 211290) - to the less so (such as the British Olympic Society, on page 20275).

Several services (including these latter two) are not actually inside the 888 area of Prestel but, because they are relevant to learning and teaching, are signposted here, which is very useful. They can be consulted even by pupils unfamiliar with the conventions of Prestel, often by following single-digit routing from within Campus 2000 or limited keywords. One way or the other this is true of every page mentioned in BEEBUG Education this month.

Nor are the information and resources within this area of Prestel restricted to any one age group: there is, for example, a very dynamic Special Needs database, SEND, beginning on page 515. And, although infants would find less than older pupils and teachers, there is material suitable for early readers.... graphics (try page 887462) and interactive stories etc. Indeed, one of the better types of service is that provided by users themselves.

A good example is called "School Time" (page 88761), provided under the aegis of one of the longer-standing IPs (Information Providers) on Prestel, School Link. Pages up at the time this month's BEEBUG Education was being written included a simple adventure game written by pupils from a school in Scarborough, where choices about dilemmas (over losing homework) encountered by many a school age pupil are dealt with by routing you through different pages in the School Time database. For example: press '1' for "tell a lie" (that you haven't lost it) or '2' if you decide to own up.

This is a simple but appealing idea and one which could be used very effectively to introduce pupils who are already familiar with text adventure games of this sort to the concept of Prestel frames. Indeed, certain GCSE and Adult Education literacy courses where students opt for a viewdata module (e.g. that taught for the RSA CLAIT 1, Computer Literacy and Information Technology) could also make good use of this and similar "games" on Prestel Education.

By a similar token, use of the educational E-Mail features would meet head on the National Curriculum requirements (English AT 2) that pupils have experience of communicating in this way. Not only is there the (now somewhat enhanced) regular Prestel Mailbox facility but also, for example, on Prestel Education (also by School Link) a service for electronic pen pals called RSVP (page 8870071). There is a mailbox directory covering specifically educational establishments on page 88804.

In common with most (though not all) of the services provided on Prestel Education, School Link is particularly good at updating its pages. Most IPs seem to respect school holidays, which is no great handicap and pupils who log on once a week in term time can expect to see new material in the majority of cases.

Continuing the tour, there is much to interest the subject specialists (an index is on page 888420) or indeed those teachers/lecturers who want to keep abreast of maths, sport, geography, home economics etc. There is also a large (separately maintained) health education section (16211).

For those responsible for administration and the "world of education" in general there is much on Prestel to meet their needs....ECCTIS, the Educational Counselling and Credit Transfer Information Service (888188), has information on 60,000 award bearing courses currently offered in colleges of further and higher education in the UK. Like several others, it is accessed through a *gateway*. There is copious information on bibliographies, correspondence courses, open and distance learning, and libraries as well as a specific "Universities Database" (page 2112). A wider careers service called Signpost can be found on page 270270.

Campus 2000 even provides an editing service for anyone wishing to promote their organisation within its pages (page 21150). There are also several 'training agencies' such as PICKUP (8881881) run for the DES, no less. Ample information on all manner of educational aids and resources exists, too. Pages covering this begin on 8885. Educational telesoftware, much of it of a very high standard and in a variety of downloading protocols exists in various places (try pages 8802 and 8803).

A most useful area of Prestel Education is the *Notice Board* area (an index to the main services is on page 888454). At the time of writing it contained, for example, news relating to school trips, American Football and career material for teachers.

Like much else, such a spot encourages us, the users, to put up information of our own and thus be part of the enterprise. Anything of interest to other pupils or staff could well be sent in: dates of events, competitions and results, visits and news of special offers to schools. Such an area also begins on page 8888811. There is also - and it would be a positive move to extend it - some room here for teacher-teacher debate of the kind that happens in the press.

Some services actually require you to join them. They operate as *Closed User Groups* (CUGs), and most of their frames are inaccessible to the general user. It would be as well to explain this to pupils, for whom the signposting to areas they cannot actually visit might be confusing or disappointing. You are also invariably warned if any page is going to carry a frame charge. It is usually only nominal anyway (usually 1, 2 or 5p).

Once you know your way around (and this is likely to take the greatest amount of time) you can go straight to your favourite pages and be reasonably sure that they are where they were when you last looked. Most decent comms software will let you assign page numbers to the BBC/Master's function keys to help here. Into the bargain, you'll need soft and hardware that can access Prestel preferably at 2400/2400 and capture frames in a buffer for later saving and printing off line to save as much money as possible. *Modem Master* from BBC Soft is one such typically suitable package.

So, having cleared the expense with the powers that be, you are recommended to look into Prestel (at local call rate for the vast majority of UK users).... there will surely be something for you and much more than you thought!

Workshop - Data Compression (continued from page 37)

```
 1070 :
 1080 DEF PROCfilltree
 1090 LOCAL d,k,n,h,i,j,t,@%
 1100 PRINT"Tree is:"
 1110 PRINT"ASC Chr Code           Freq"
 1120 FOR k=0 TO 511:h=node%(k)
 1130 IF h=-1 THEN 1250
 1140 i=0:j=1:t=dad%(k):x=0:REPEAT
 1150 IF t<0 THEN t=-t ELSE x=x+j
 1160 t=dad%(t):j=j*2:i=i+1:UNTIL t=0
 1170 code%(h)=x:len%(h)=i
 1180 PRINT h;TAB(5);
 1190 IF h<127 AND h>31 THEN PRINTCHR$h;
 1200 PRINTTAB(8);:d=2^(len%(h)-1)
 1210 FOR n=1 TO len%(h)
 1220 PRINT SGN(x AND d);:d=d DIV 2
 1230 NEXT
 1240 PRINTTAB(23);count%(k)
 1250 NEXT:ENDPROC
 1260 :
 1270 DEF PROCxlate
 1280 INPUT"Output file name:"m$
 1290 g=OPENOUTm$
 1300 IF g=0 THEN PRINT"Unable to open o
utput file":END
 1310 f=OPENINn$:T%=0
```

```
 1320 out%=0:fac%=128:REM init buffer
 1330 REPEAT x=BGET#f
 1340 T%=T%+len%(x)
 1350 PROCsend(code%(x),len%(x))
 1360 UNTIL EOF#f
 1370 PROCsend(0,7):REM empty buffer
 1380 PRINT'"Compression factor "T%/EXT#
f/8
 1390 CLOSE#f:CLOSE#g:ENDPROC
 1400 :
 1410 DEF PROCsend(c%,n%):REM send c% in
 n% bits
 1420 LOCAL i%:d%=2^(n%-1)
 1430 FOR i%=1 TO n%
 1440 PROCsend1(SGN(c% AND d%))
 1450 d%=d% DIV 2
 1460 NEXT
 1470 PRINT".";
 1480 ENDPROC
 1490 :
 1500 DEF PROCsend1(c%):REM sent 1 bit
 1510 LOCAL @%:PRINTc%;
 1520 out%=out%+c%*fac%
 1530 IF fac%=1 THEN BPUT#g,out%:out%=0:
fac%=256
 1540 fac%=fac% DIV 2:ENDPROC
```

An Improved Route Planner (continued from page 25)

```
stm%>59 THEN 3230
 3250  FL%=1:PF%=1:X=OPENUP(f$):PROCproc
ess
 3260  ENDPROC
 3270  :
 3500  DEFPROCcat
 3510  LOCAL A%,B%,C%
 3520  R%=0:buffer%=&7000:FOR dr%=0 TO 2
 STEP 2
 3530  ?&70=dr%:!&71=buffer%:?&75=3:?&76
=&53:?&77=0:?&79=34
 3540  X%=&70:Y%=0:A%=&7F:CALL&FFF1
 3550  nof%=(buffer%?&105)/8
 3560  C%=buffer%+8
 3570  FOR A%=15.5*dr% TO (nof%-1+15.5*d
r%)
 3580  IF ?C%<>32 name$(A%)=""
 3590  FOR B%=0 TO 7:name$(A%)=name$(A%)
+CHR$((C%?B%)AND127):NEXTB%
 3600  IF ASC(name$(A%))=0 GOTO 3670:ELS
E g$=":"+STR$(dr%)+"."+LEFT$(name$(A%),7
```

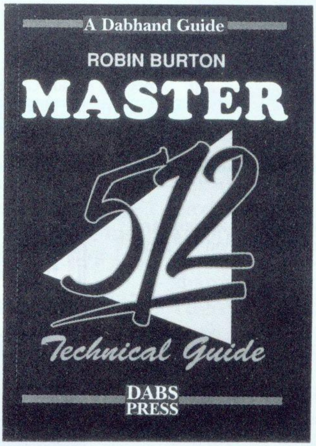
```
)
 3610  X=OPENUP(g$)
 3620  check$=FNupper(FNgetstring):IF ch
eck$="M" OR check$="K" GOTO 3640 ELSE 36
30
 3630  name$(A%)="":GOTO3670
 3640  name$(R%)=g$
 3650  PROCbegend
 3660  beg$(R%)=beg$:end$(R%)=end$:R%=R%
+1
 3670  CLOSE#0:C%=C%+8:NEXT A%:NEXT dr%
 3680  ENDPROC
 3690  :
 4000  DEFPROCbegend
 4010  LOCAL A%
 4020  FOR A%=0 TO 6
 4030  remove$=FNgetstring
 4040  NEXT A%
 4050  beg$=FNgetstring:end$=FNgetstring
 4060  ENDPROC
```

Master 512 Technical Guide

Reviewed by Bernard Hill

Book	**A Dabhand Guide: Master 512 Technical Guide**
Author	**Robin Burton**
Publisher	**Dabs Press 22 Warwick Street, Prestwich, Manchester M25 7HN. Tel. 061-773 8632**
Price	**£14.95 (£19.95 with programs disc)**

It is difficult to believe that it is nearly two years since I reviewed Chris Snee's *Master 512 User Guide* for BEEBUG (see Vol.8 No.1), and we expected the *Master 512 Technical Guide* to be available that summer. Well here it is at last, but was it worth the delay?



This book is twice the size of its companion volume mentioned above. Running to 416 pages, there are 11 chapters and 12 appendices, the latter occupying well over half the book! But maybe that in itself gives a feeling for its content. The 'main' part of the book consists of a description of the whole interlocking 6502-80186 twin system, and the appendices are mainly reference material. The first part of the book is constructed as follows:

1. Chapters 1 and 2 form a short introduction, system overview and a 12-page look at the 80186 itself. This is a brief and simple description of the processor, its registers, memory segmentation and addressing techniques.

2. Chapters 3 to 5 outline the way in which the 80186 monitor program (which appears when you do a 'soft Break') works and responds, and also details the system initialisation and boot procedures, the downloading of the communication program 6502.SYS which will sit in the BBC, and the way in which messages are sent between the two processors.

3. Chapter 6 begins a more technical set of chapters about the access points which DOS has to the various BBC OS calls (OSASCI, OSFILE, OSBYTE and the like). These access points are known in 80186 terms as 'software interrupts', and Chapter 6 describes and lists the correspondence between them and the OS calls. Chapter 7 outlines the DOS philosophy of running a program (.EXE, .COM, .CMD and .RSX files - the latter being quite new to me), and Chapter 8 is an overview of the software interrupts. Chapter 9 is a little easier to read and outlines the structure of the various disc formats which are used by DOS.

4. Chapters 10 and 11 complete the first part of this technical guide and are full descriptions of the MOVE and EDLIN programs whose function and operation was only briefly indicated in the companion volume.

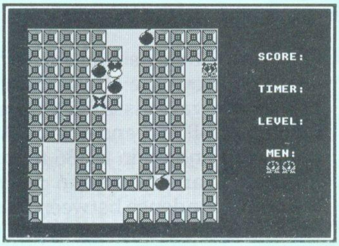
THE APPENDICES

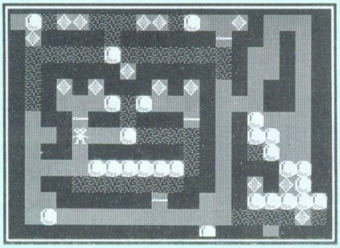
The software interrupts which I referred to above are identified by a number and a subnumber. As will be very familiar to MS-DOS programmers, Interrupt 21 (hex) is the principal system access point from an assembly program to the operating system (analogous to the JSR &FFxx's of the 6502). Appendices A to C contain nearly 100 pages of details about this and other interrupts, but new to those fluent in MS-DOS will be the function 224 calls which are in origin CP/M 'BDOS' operations and take the place of the PC's BIOS calls.

Appendix D is a list of disc formats and Appendices E and F contain a fully commented (6502) assembly source code listing of the Tube Host Code and 6502.SYS, printed by permission of Acorn, (which was also given for the back-page foldout circuit diagram of the full 512 co-processor).

Thus far we have 330 pages packed full of information, but the remainder of the book is

Arcade Games





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

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

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

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

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

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

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

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

Beebug Arcade Games Disc **£5.95** **+ 60p p&p**
Stock Codes **PAG1** (5.25" DFS 40/80T disc)
PAG2 (3.5" ADFS disc)

Board Games

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

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

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

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

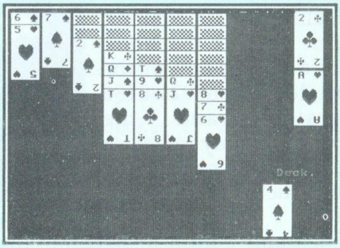
Cribbage - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

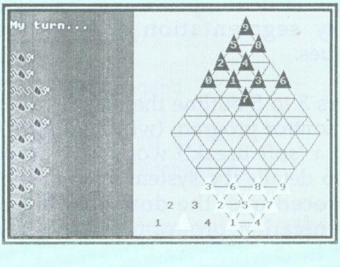
Twiddle - a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern.

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

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

Beebug Board Games Disc **£5.95** **+ 60p p&p**
Stock Codes **PBG1** (5.25" DFS 40/80T disc)
PBG2 (3.5" ADFS disc)





Auto-Start for the RFS

by Kai S Ng

Jon Keats' three articles on the ROM Filing System (Vol.8 Nos.8 & 9, Vol.9 No.3) provided an excellent insight into a little-used system. The article presented here supplies an enhancement to one of Jon Keats' listings, by providing an auto-start facility of the type commonly used on discs. It works by detecting when "R" and Break are held down together. If so, the RFS is engaged and any boot file therein is executed.

THE UPGRADE

The listing given here is intended to be added to the program *RFShead* listed on page 27 of Vol.8 No.8. Lines 1172 and 1173 detect if the "R" key is pressed during a Break. If so, the text in line 1179 is sent to the keyboard buffer to be executed when the machine is ready. This consists of the command **ROM*, followed by **EXEC !Boot*. The !Boot file can be conveniently created using *BUILD. Note that this command is not recognised by the RFS, so the !Boot file should first be created on a disc and transferred to the RFS using the RFSload program from Vol.8 No.9.

```
1011 osbyte=&FFF4
1101   CMP #3 :BEQ boot
1171 .boot  \service call 3
1172   PHA :TXA :PHA :TYA :PHA :LDA #&79
:LDX #(&33 EOR&80) :JSR osbyte
1173   CPX #0 :BEQ boot_q :LDA &80 :PHA
:LDA #0 :STA &80 :LDA #138
1174 .boot_loop
1175   LDX &80 :LDY bootline,X :BEQ boot_p
1176   LDX #0 :JSR osbyte :INC &80 :JMP
boot_loop
1177 .boot_p PLA :STA &80
1178 .boot_q PLA :TAY :PLA :TAX :PLA :RTS
1179 .bootline EQUS "*ROM" :EQUB 13 :EQUS
"*EXEC !BOOT" :EQUW 13
```

Master 512 Technical Guide

quite different. Appendix G outlines hardware projects to upgrade the processor speed by 20%; to construct a 512K RAM expansion board and to add a hard disc unit. Readers will forgive me if I did not attempt my own upgrades in assessing this book!

The remaining sections of the book consist of a list of third party products, both hardware and software (BEEBUG gets a mention for its 512 Forum), a glossary, details of the optional accompanying disc and the obligatory list of other Dabs Press books.

TARGET AUDIENCE

I found the book fascinating: Robin's admiration for Acorn's ingenuity shows through on every page, but in reading it I found myself continually asking who the book was written for. In comparison with (say) the readers of the BBC Advanced User Guide there can be only a handful of people who will write applications for the 512. Maybe more will construct a 512K expansion board but I puzzled for some time over who would value the book. But in looking at the contents of the accompanying disc I began to see a larger target audience.

On this disc can be found some time-limited demonstration copies of Essential Software products (reviewed in Vol.8 No.5 and Vol.9 No.1); a copy of BBC Basic (512) which Robin has previously distributed, and an excellent shareware assembler/debugger called A86/D86. There are also some demonstration programs with source code for you to assemble and investigate yourself.

So maybe here lies the clue to an extended readership: like all BBC Basics this one contains a built-in assembler, so with the stand-alone assembler and sample programs you have an excellent start into the world of 512 assembly programming. In BBC terms, buying this book and disc is the equivalent of buying an Advanced User Guide and a copy of BEEBUG's Exmon. And a large number of BBC programmers started out in exactly that way.

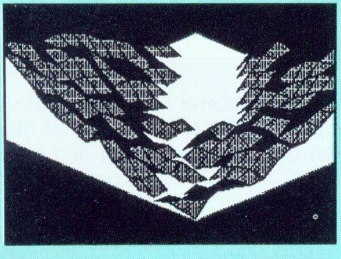
CONCLUSIONS

In terms of information content, the *512 Technical Guide* has no equal. In terms of value in the PC world you could expect to pay around £40 for this sort of information. If you're interested in the workings of the 512 at all then this book at £15 (or £20 with disc) is superb value for money.

Well done Robin and Dabs. Another winner.

Best of BEEBUG Software

Applications II Disc



Crossword Editor - for designing, editing and solving crosswords

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

3D Landscapes - generates three dimensional landscapes.

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

Running Four Temperatures - calibrates and plots up to four temperatures

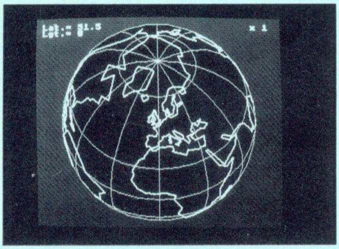
Julia Sets - fascinating extensions of the Mandelbrot set

Foreign Language Tester - foreign character definer and language tester

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

Share Investor - assists decision making when buying and selling shares.

Applications I Disc



Business Graphics - for producing graphs, charts and diagrams

Video Cataloguer - catalogue and print labels for your video cassettes

World by Night and Day - a display of the world showing night and day for any time and date

Phone Book - an on-screen telephone book which can be easily edited and updated

Page Designer - a self contained page-making package for Epson compatible printers

Personalised Letter-Headings - design a stylish logo for your letter heads

Appointments Diary - a computerised appointments diary

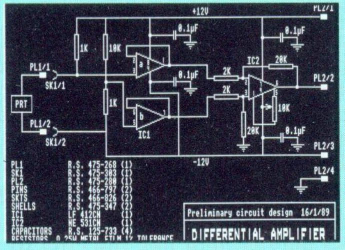
Mapping the British Isles - draw a map of the British Isles at any size

Selective Breeding - a superb graphical display of selective breeding of insects

The Earth from Space - draw a picture of the Earth as seen from any point in space

Personalised Address Book - on-screen address and phone book

ASTAAD



```
PL1          R.S. 475-268 (1)
SK1          R.S. 475-303 (1)
PL2          R.S. 475-280 (1)
PINS         R.S. 466-797 (2)
SKTS         R.S. 466-826 (2)
SHELLS       R.S. 475-347 (2)
IC1          LF 412CN     (1)
IC2          NE 531CN     (1)
CAPACITORS   R.S. 125-733 (4)
RESISTORS  0.25W METAL FILM 1% TOLERANCE
```

Preliminary circuit design 16/1/89

DIFFERENTIAL AMPLIFIER

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

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

General Utilities Disc

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

° **Master series only.**

† **Requires sideways RAM.**

EDIKIT ROM

An indispensible utility ROM for all Basic programmers, containing the following commands:

***FTEXT** (find text) ***FBASIC** (find Basic) ***FPROCFN** (find procedure/function) ***LPROC** (list procedure) ***LFN** (list function) ***LFROM** (list 8 lines of a program) ***RTEXT** (replace text) ***RBASIC** (replace Basic) ***SYSINF** (system information) ***VARLIST** (list program variables) ***FKDEFS** (function key definitions)

Incorporating the updated Basic Booster utilities:

SUPER SQUEEZE **PARTIAL RENUMBER** **PROGRAM LISTER**
RESEQUENCER **SMART RENUMBER** **TEXTLOAD AND TEXTSAVE**

	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407A	£5.95	**ASTAAD** (3.5" ADFS)	1408A	£5.95
EDIKIT (EPROM)	1451A	£7.75			
EDIKIT (40/80T DFS)	1450A	£5.75	**EDIKIT** (3.5" ADFS)	1452A	£5.75
Applications II (80 track DFS)	1411A	£4.00	**Applications II** (3.5" ADFS)	1412A	£4.00
Applications I Disc (40/80T DFS)	1404A	£4.00	**Applications I Disc** (3.5" ADFS)	1409A	£4.00
General Utilities Disc (40/80T DFS)	1405A	£4.00	**General Utilities Disc** (3.5" ADFS)	1413A	£4.00

Please add p&p - 60p for the first item and 30p for every additional item.

BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Tel. (0727) 40303 Fax. (0727) 860263

HINTS *and tips* HINTS *and tips* HINTS *and tips* HINTS *and tips* HINTS *and tips*

Hints and tips on almost any subject relating to the BBC micro and Master series are always welcome, and we pay £5 for each one published.

FINDING VARTOP

David Stevens

You sometimes need to find the address of the first free byte above a program's variable storage area. This information is store in bytes &02 and &03 of zero page. However, if you attempt to access this information as a hex number, any leading zeros will be left out. Thus to access the value correctly in hex use:

```
vartop=~(?2+?3*256)
```

You can either use this in immediate mode after running a program, to determine the extent of its variable storage area, or from within a program while it is running.

PLUS/MINUS IN INTERWORD/VIEW

G.P.Stenning

A simple way to produce a 'plus-or-minus' sign when using Interword is to type '+' and then to relocate the cursor to that position. Mark it using Ctrl-Z followed by Shift-Underline. Move the cursor off the resulting '+̲' and continue as required (e.g. 900 +̲66). Note that a space is essential before the 'plus-or-minus' if the previous word/number is not also to be underlined. The same result can be achieved in View by preceding and following the '+' with Shift-f4 to underline the character (but the extra space is not then needed).

COLOUR ORDER

Hugh O'Byrne

To arrange the colours on the BBC micro (particularly in mode 2) in order of grey shades, the colours are best expressed in binary as FGBR, as red gives the darkest shade of grey, blue the next and green the brightest (F is the flash bit). However, the colours are normally represented as FBGR which gives a different ordering.

Here is a function which puts the colours in sequence as grey shades:

```
1000 DEF FNsh(A%)=(A% AND 8)+(A% AND
6)DIV2+(A% AND 1)*4
```

FNsh(0), for example is black, and the shades increase through to FNsh(7) which is white. Thus everywhere in a program there is a statement of the form:

```
GCOL 0,X
```

this could be replaced by one of the form:

```
GCOL 0,FNsh(X)
```

This way a screen dump can be saved without having to change the colours each time you reload it.

Finally, here is a procedure which works with a normal palette and gives 15 shades using ECF patterns:

```
1000 DEF PROCcol(A%)
1010 D%=FNsh(A% DIV 2):E%=FNsh((A%+1)DIV2)
1020 IF D%=E% THEN GCOL 0,D%:ENDPROC ELSE
B%=(D% AND 1)+(D% AND 2)*2+(D% AND 4)*4:
C%=(E% AND 1)+(E% AND 2)*2+(E% AND 4)*4
1030 V%=B%+C%*2
1040 VDU23,2,V%,V%,V%,V%,V%,V%,V%,V%
1050 GCOL16,0:ENDPROC
```

MACHINE CODE HINTS

David Holton

If you start your assembly program with the square bracket on a separate line from the OPT statement, or if you separate them by a colon, then the value of P% is displayed on each pass. If you wish to suppress P%, say because you want to avoid a ragged start to a Basic program containing assembler, just put the bracket next to the OPT statement or separate them by a space, thus:

```
    [OPT opt%
or: [ OPT opt%
```

If you then want to see the final value of P%, end with:

```
]:[:]
```

which effectively re-enters the assembler with [:.

To check assembly language programs write a Basic line at the end to test that the value of P% is correct. It'll catch some errors at least.

RISC USER

The Archimedes Magazine & Support Group

Now in its fourth year of publication, Risc User continues to enjoy the largest circulation of any magazine devoted solely to the Archimedes range of computers. It provides support for all Archimedes users at work (schools, colleges, universities, industry, government establishments) and home.

Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines.

A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer to enthusiasts and professionals at all levels.

And here are some items covered in the most recent issues of RISC User:

WATCHDOG ANTI-VIRUS
An improved version of this very popular utility for protecting your hard disc from viruses.

MAGPIE
A review of this excellent information organiser from Longman Logotron.

DESKTOP ZOOM
A Wimp application which allows you to magnify any part of the Desktop.

HARD DISC ARCHIVER
An excellent utility for selectively backing-up the contents of your hard disc onto floppies.

WP/DTP
A regular column which offers advice on using different DTP and WP packages.

FIRST LOOK AT BEZIER CURVES
Two-part series on this mathematical method for drawing smooth curves and surfaces.

A MULTI-TASKING WIMP DEBUGGER
A multi-tasking application which allows you to debug Wimp based programs while testing and running them.

PRINTER AND SERIAL BUFFER FOR THE ARC
A variable sized buffer which can be configured either as a printer buffer, or a serial buffer.

PROGRAMMER'S WORKSHOP
A major series covering all advanced aspects of the Archimedes and incorporating the Assembler Workshop. This month: understanding E-format discs.

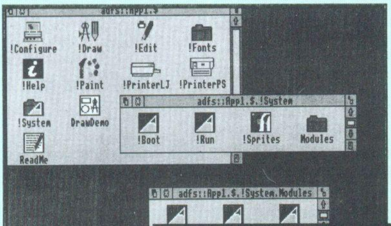
MASTERING THE WIMP
A major and very popular series on the Wimp programming environment. The most recent installment is devoted to Using Templates.

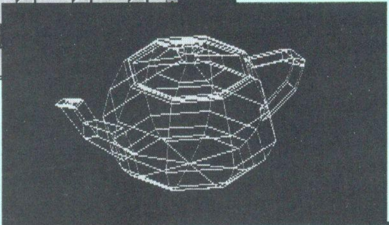
INTO THE ARC
A regular series for beginners. The latest article explores some aspects of using colour on the Archimedes.

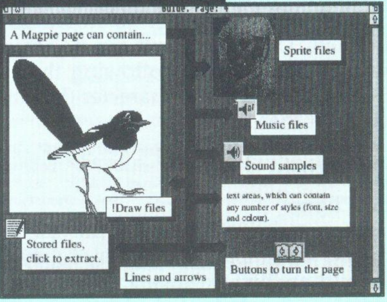
INTRODUCING C
A wide ranging series on the major programming language C.

ARCADE
A round-up of the latest games for the Archimedes:

ArcPinball,
Iron Lord,
The Wimp Game,
Professor Mariarti,
Twin World.






A Magpie page can contain...
Sprite files
Music files
Sound samples
text areas, which can contain any number of styles (font, size and colour).
!Draw files
Stored files, click to extract.
Lines and arrows
Buttons to turn the page

As a member of BEEBUG you may extend your subscription to include RISC User for only **£9.10** (overseas see table).

Phone your instructions in, or send a cheque/postal order to the address below. Please quote your **name** and **membership number.** When ordering by Access, Visa or Connect, please quote your card number and the expiry date.

SUBSCRIPTION DETAILS

Destination	Additional Cost
UK,BFPO &Ch Is	£ 9.10
Rest of Europe and Eire	£14.00
Middle East	£17.00
Americas and Africa	£19.00
Elsewhere	£20.00

RISC User, 117 Hatfield Road, St Albans, Herts AL1 4JS, Telephone (0727) 40303, FAX (0727) 860263

POSTBAG

POSTBAG

ROM IMAGE LOADER FOR SECOND PROCESSORS

The sideways ROM image loader program by John Lasruk in BEEBUG Vol.9 No.6 can be made compatible with all second processors by altering line 1620 to read:

```
1620 OSCLI "SAVE "+out$+" 2F3A "+top$+
" FFFF2F3A FFFF2F3A"
```

The extra parameters in the *SAVE Command set the reload and execution addresses to &2F3A in the I/O processor's memory, thus ensuring that it executes in the I/O processor's memory and is able to transfer the ROM image into a sideways RAM.

R.C.Smith

MODIFIED ADFS DIRECTORY EXAMINER

Can you please supply a solution to the puzzle posed on page 14 of BEEBUG Vol.9 No.4 on rewriting lines 180 and 2060. I haven't a clue how to change these lines to work on my model B with the ADFS.

T.D.Parsons

A number of readers have asked for help with this. The original program listing used a feature of Basic only found on the Archimedes range (Basic IV and later). This is the construction:

```
ON <expression> PROCa, PROCb, . . .
```

To recode the original lines, replace line 180 by a series of lines:

```
180 IF ASC(key$)-245=1 THEN PROCdisc
181 IF ASC(key$)-245=2 THEN PROCat_command
```

etc.

and line 2060 by a similar sequence of lines:

```
2060 IF key$<>CHR$(27) AND ASC(key$)-252=1
        THEN PROCtype
2061 IF key$<>CHR$(27) AND ASC(key$)-252=2
        THEN PROCdump
```

etc.

This may not be the most concise or efficient way to recode the lines in question, but does keep closely to the format of the original, and should therefore serve as a guide for any similar conversions which might be required.

PROBLEMS WITH THE NEW MASTER ROM

With reference to Derek Gibbon's article in BEEBUG Vol.9 No.6, I have met with one or two problems since installing the new Master System ROM. However, I cannot say with certainty whether they are due to the new ROM or some other cause.

In ROM position 3 I have fitted the BEEBUG Master ROM. The *P command calling up the control panel correctly shows Spellmaster in ROM positions 6 & 7, whereas the *ROMS command only shows Spellmaster in position 6. The *MERGE command is non-operative, and there is no error message. All my files are in ADFS format, but to copy files it now appears essential to use the *FCOPY command and to prefix both source and destination details with 'A:'. I also find problems in using the ADFS version of Sharemaster (Synergy Software).

In general I prefer the new system ROM for its speedier ADFS file handling and mathematical computation, but can anyone throw any light on the problems described above?

David Holme

Further comment and feedback on the consequences of using the new Master system ROM would be appreciated, and where appropriate useful information will be reproduced in these pages.

THANK YOU

My sincere thanks to you and your staff for your assistance. Due to your efforts in publishing in Personal Ads in BEEBUG a note of my son's need for 'Aviator' to run on a Master we were very kindly supplied with that very thing by another BEEBUG member within days of publication of the December issue.

It's pleasing to see how BEEBUG members and Beeb users in general help one another.

Charles Butler

Note: Mr Butler's son is disabled following an accident and was desperate for a disc version of Aviator (no longer available new), being unable to handle his previous cassette version. We suggested he try our personal ads columns with the splendid result outlined above.

Personal Ads

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

We also accept members' Business Ads at the rate of 40p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 15th of each month.

Master 512 co-processor plus Watford Co-Pro Adaptor, Dabs Master 512 User Guide and disc, mouse, Tull 512 mouse driver, GEM, Baksoft Doscopy £200, British Micro Graphpad Mk2 £40. Tel. (0707) 54311 eves.

Inter Mega3 ROM £60, Interbase ROM £40, Apricote Account Book V3 + invoice V2, ADFS discs £30, all as new, boxed and still with full documentation. Tel. 071-508 1737.

Oxford Pascal ROM for model B only. Includes book and 40T disc in presentation folder, this package has never been used £20. Tel. 081-654 3733.

512 co-processor with mouse and software £95 o.n.o. 'B' ATPL ROM board with 16k battery backed RAM £25 o.n.o. 'B' Floppywise+ disc utility ROM, Lisp (disc), Gremlin monitor/Debug ROM, chauffeur mouse-driver package, DFS 80T Masterfile II database, Elite, Hitchhiker's guide (suitable for 512) all £12 o.n.o. 'B' Microtext (disc) package, BCPL (ROM) package with direct access & calc. £25 each o.n.o. 3x5.25" 40T full height drives £15 each o.n.o. Hitachi 12" green screen monitor £30 o.n.o. Acorn User collection, July/Aug '82 (no.1) to May '86 £30 o.n.o. Tel. (0273) 605339 after 8.30pm.

Printmaster ROM (BBC/Epson) £12, Wordwise (not 'Plus') £12, Beebugsoft Spellcheck discs for Wordwise £8, all complete with manuals. Tel. (0226) 762450.

Bargains! Command + Magic Modem (auto-dial/answer) £65, E-Type £6, Ricochet £5, Play It Again Sam 13 & 8, AMX Pagemaker (magazine designer) £20! Command £20, Speech £3, Accelerator (Basic compiler) £30, Acorn User 7 disc compilation of games and utilities £15. Tel. (066479) 525.

Master 128, complete with Reference manuals 1&2, View and Viewsheet manuals, two ROM cartridges £300, Philips green screen hi-res. monitor £40, Cumana CD800S dual 40/80T D/S switchable disc drive with own PSU £120, AMX Superart and mouse £25, Viewplot £12, Viewstore £25, Mega-3 ROM package - one ROM containing Intersheet, Interchart and Interword £50, Datachart 1223 modem with MNET ROM £45, Watford Electronics co-pro adaptor £30, W.E. ROM Manager £10, Printmaster and Printer monitor ROM £7 each. All excellent and complete (upgrading). Tel. 051-606 0289.

BBC B issue 7, Acorn ADFS DFS, Watford 32k Shadow RAM, ATPL ROM board with 2x8k S.R. chips, 40T disc drive, Sanyo green mono monitor, Mini Office II, Elite, Studio 8 plus games disc £275. Could deliver in Midlands. Tel. (0384) 377710.

Panasonic KXP1124 printer, good condition £80. Tel. 081-449 6681 eves.

512 add-on board for model B or Master, in external Acorn case with own PSU, 4 supplied discs, mouse and manual. Also a Dabhand Guide and 13 Dabs Press shareware discs £200. Tel. 021-382 3606.

BBC Teletext system for BBC B or Master, boxed complete with User Guide and Advanced ROMs versions 3 & 3.1 £30 to clear. Tel. 065671 8187.

BBC model B issue 7, 1770 DFS, ADFS, 40T drive, ROM/RAM board, 16k sideways RAM, plenty of discs £275. Tel. (0925) 755139.

RX-8 plus APT-1 weather satellite decoding module, as new, cost £320 accept £160. Tel. 091-548 5586.

BBC B issue 4 with Watford 40/80T DS/DD 3.5/5.25" disc drive and Watford Solderless ROM/RAM expansion board with battery back-up. Software includes Logotron Logo, Mini Office II and many games, also joystick £250 o.n.o. Also BBC Master 128, Cumana disc drive, daisy wheel A3/A4 printer, usual software plus Interword, Interchart, Intergraph many games books and magazines £350 o.n.o. Will sell items separately. Tel. (0462) 676761.

WANTED: Floppy disc version of Wordperfect 4.2, capable of running on Master 512, with documentation. Any offers? Current versions need more memory. Tel. 071-228 6726 day.

Cumana double 5.25" disc drive 40/80T switchable own PSU, plus about 40 discs. Only £65. Buyer collects or pays postage. Tel. (0328) 864177.

Genie cartidge £30, C.C. Mega3 ROM £35, AMX Pagemaker £15, AMX Superart £10, Novacad £10, Mos+ Disc £4, Elite Disc (for BBC/Master 128) £5, Wapping Art Disc £8, WYSIWYG + ROM £10, P.I.A.S. 1&2 (for Master compact) £8, or £100 the lot. Tel. (0332) 572009.

Acorn Prestel adaptor for BBCs and Master, complete with ROM manual and keystrip, in good condition. Excellent for all Viewdata systems £45, postage and packing included. Tel. (0977) 704083.

BBC B issue 7 with DFS £170, Cumana twin 40T disc/Int PSU £60, BEEBUG Vol.1-8 in binders £70, BEEBUG tapes Vo. 2-5 all £24, Wordwise Plus ROM, Exmon II ROM, Masterfile II disc, all manuals, many books, tapes. Offers invited. Tel. (0529) 302438.

BBC B 1.2 plus, datarecorder, Cumana disc drive, joysticks, B&W Ingersoll television, Amber 2400 printer and numerous tapes, discs and magazines etc. £450 o.n.o. Tel. (0344) 422961.

£5 for any kind person who can supply a copy of the instructions for Cambridge Red Boxes for BBC computer. Tel. (0992) 556075 day.

Free desk with Master 512, twin DS 40/80T + PSU disc drive, Microvitec colour and Ferguson mono monitors, FX-80 printer, manuals, software £650. BEEBUG, Micro and Acorn User magazines complete - negotiate. Buyer collects. Would split 512 board etc. Tel. (0242) 674470.

Archimedes 310, twin DD, colour monitor and software £600. Tel. (0743) 248107.

Maplin MF100 multifunction counter, 100MHz, perfect condition £75, Teletext decoder (Acorn) £38, Electron plus one (printer port, analogue port, twin cartridge port), with datarecorder, books and tapes £48, AMX mouse (grey/red) £15, BEEBUG Vol.1 complete £10, BEEBUG magazine tapes covering Vol.5, No.3 to Vol.9 No.3 £6 per vol. Acorn cartridges, unused £5 each, Watford EPROM Eraser, takes upto 16 EPROMs, unused £25. Tel. 081-864 0309.

BBC M128 in excellent condition, plus single 3.5" DSDD disc drive, tape and ROM cartridges, price includes some ROM/disc based software, all programming/user manuals, leads, original packaging only £345 o.n.o. (upgrading to Arc.) Tel. (0283) 31403 anytime.

Epson LQ2500 24 pin DM printer, first class print quality with 5 built in print modes/fonts, very fast - 400 cps print speed, hardly used and boxed as new with 2 new ribbons. Cost over £900 new, sell for £425 o.n.o. Tel. 081-967 4401.

WANTED: Acorn User/Micro User magazine discs 5.25" pre 1989. Tel. (0482) 707099.

M128 as new £250, Micronix twin/double sided 40/80T disc drive £100, Philips green screen monitor working fine but no documentation £20, Morley AA Expansion board £25, ROMs - Interword £20, Interbase £25, Spellmaster £20, Instant Mini Office £25, Edikit £3, Dumpmaster £10, Master Ref. manuals £5 each, The Advanced User Guide for the BBC B £3, Bruce Smith's BBC Micro ROM book £5, Getting Started in BBC Basic £1, 30hour Basic (BBC std. edition) £2, mix and match what you want (postage extra). Tel. 091-581 9989.

ATPL Sideways ROM board for model B computer including 16k RAM £25, W.E. 'Adder' EPROM programmer £35, both items with full documentation. Tel. (0481) 56266 after 6pm.

Morley AA board £25, Interword £25, Interbase £25, Intersheet £20, Interchart £20, Spellmaster £30, all boxed with manuals, Morley Smart cartridge £25, BEEBUG Master ROM £15, Wapping Editor (inc. mouse & mat) plus Art disc, Utility disc £50, Beeb Handscan plus Firmware £65. Will consider selling the lot for £250 o.n.o. Tel. (0293) 542288 eves or wk/ends.

Aries B32 £40, B12+RAM £25, Wordwise plus, Publisher £20 each, CP-ROM2, Dumpout3, Printwise, Studio8 £10 each, all boxed with manuals. Tel. 03632 2316.

BBC B issue 4, fitted with Watford DFS, Viglen 5.25" single disc drive, Watford Dumpout3 and Wordwise Plus ROMs, original data recorder, welcome tape and all manuals £140. Tel. (0494) 712457.

WANTED: Acorn Atom computer in good working condition. Tel. 091-374 3631 day or 091-372 1797 eves.

Dr Dos plus manuals, System Guide, Device Drivers Support and Programmer's Reference Guides for both CP/M and MS DOS - ideal for writing you own programs. Offers? Tel. 091-285 0097 eves.

Cumana twin 5" double sided disc drive, with built in PSU, switchable to 40 or 80T discs (CD800S), perfect condition £110 o.n.o. Tel. (0283) 62973.

Morley teletext adaptor with ATS ROM £40, BEEBUG Vols. 1-1 to 9-8 £50, W.E. NLQ ROM for Epson £10, Interbase £25, Interword £20, Clares BROM Plus £15, Pen Friend 2 £10, Clares Fontwise Plus and editor £15, Design 7 screen designer £5, Advanced disc investigator (ROM) for challenger 3 disc drive £10, Genius serial mouse for IBM PC £15. Tel. (0276) 35228.

WANTED: Disc filing system User Guide by Acorn (1770 DFS), also colour monitor med. res. Tel. 081-550 8294.

Viglen 28Mb hard disc with PSU and cables, suit BBC Master system £250. Epson JX80 colour printer with several spare ribbons £200. Acorn 2nd Processor (6502) plus manual £25. Tel. 051-334 8933 eves.

WANTED: Please could anyone kindly let me have a copy of the software supplied with the Trak-Ball sold by Datapen Microtechnology. My copy has become corrupted and the firm seem to have gone out of business. Tel. (0206) 262765.

5.25" Acorn Lisp, Viewspell, Viewplot, Microtext, 3.5" View Professional and Viewsheet £10 each (please note some manuals but no ROMs), 40/80T Cumana disc drive with PSU £90, 40T Cumana disc drive £40. Tel. 04867 80632.

Hybrid Music 5000 system, with programmer's manual, user manual and IO albums (5.25") £75 o.n.o. Tel. (0234) 271591 anytime.

WANTED: Welcome disc for Master Compact or a copy, blank disc supplied. Tel. (0424) 813794.

Z88 personal computer, complete with 32k RAM pack, mains adaptor, 4 rechargable batteries and recharger BBC/Z88 link package (disc and lead), manual, original box and leather-look carrying case (will also take several sheets of paper) £125 or very near offer (current price of this package over £200). Tel. (0344) 861988 9-5pm wk/days.

Master 512 £100, co-pro adaptor £20, AMX Stop Press £25, incl. mouse. Tel. (0372) 740678.

WANTED: Heritage - Genealogical package. Tel. (0252) 715646.

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

BEEBUG		BEEBUG & RISC USER
£18.40	1 year (10 issues) UK, BFPO, Ch.I	£27.50
£27.50	Rest of Europe & Eire	£41.50
£33.50	Middle East	£50.50
£36.50	Americas & Africa	£55.50
£39.50	Elsewhere	£59.50

BACK ISSUE PRICES (per issue)

Volume	Magazine	5"Disc	3.5"Disc
5	£1.20	£4.50	£4.50
6	£1.30	£4.75	£4.75
7	£1.30	£4.75	£4.75
8	£1.60	£4.75	£4.75
9	£1.60	£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.

POST AND PACKING

Please add the cost of p&p when ordering individual items. See table opposite.

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

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 860263
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: Alan Wrigley
Technical Assistant: Glynn Clements
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

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

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

In all communication, please quote your membership number.

ISSN - 0263 - 7561

Printed by Arlon Printers (0923) 268328

Magazine Disc

March 1991
DISC CONTENTS

CYCLIC FUNCTIONS - this program, based on the popular Spirograph, draws numerous cyclic shapes such as circles, ellipses, cardioids, deltoids and many more.

A TREASURER'S COMPANION - a program which allows you to organise membership records of any club, keep track of payments and do mailings to members. Three sample data files with names and addresses are also provided.

OUTFONT: AN OUTLINE FONT DESIGNER - this outline font designer complements the very popular program for printing spline text (Vol.9 No.2) which is also included on the disc.

AN IMPROVED ROUTE PLANNER - the complete route planner from Vol.7 No.6, incorporating the amendments from this month plus an example data file Cornwall Route.

AN ADFS/VIEW UTILITY FOR THE MASTER - a program which helps you manage your View text files more flexibly and more efficiently.

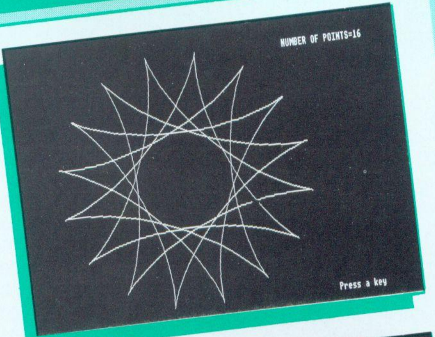
BEEBUG WORKSHOP: DATA COMPRESSION - a program demonstrating Huffman encoding for compressing data, so that it can be stored in less disc space.

BEEBUG FUNCTION PROCEDURE LIBRARY - a Basic program containing all the functions and procedures from this month's article.

PRACTICAL ASSEMBLER (8) - an enhanced file copying utility written in assembler for sideways RAM.

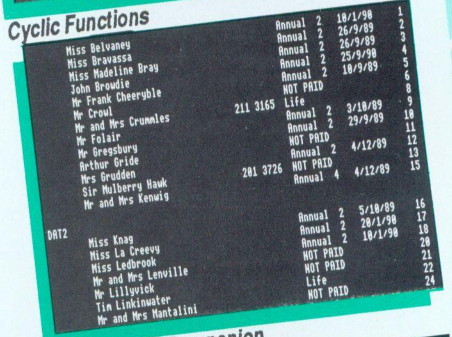
AUTO-START FOR THE RFS - an enhancement to the original RFS listings (Vol.8 Nos. 8,9, Vol.9 No.3), providing an auto-start facility.

MAGSCAN DATA - bibliography for this issue (Vol.9 No.9).



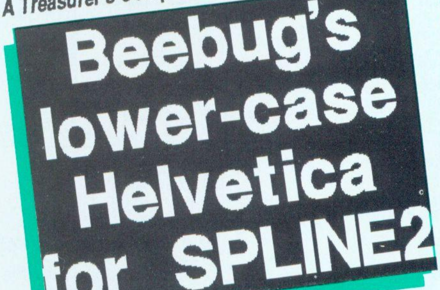
```
                                   NUMBER OF POINTS=16

                                             Press a key
```

Cyclic Functions



```
                                        Annual  2  10/1/90    1
                                        Annual  2  26/9/89    2
    Miss Belvaney                       Annual  2  26/9/89    3
    Miss Bravassa                       Annual  2  25/9/90    4
    Miss Madeline Bray                  Annual  2  10/9/89    5
    John Browdie                        Annual  2             6
    Mr Frank Cheeryble                  NOT PAID              8
    Mr Crowl                   211 3165 Life                  9
    Mr and Mrs Crummles                 Annual  2  3/10/89   10
    Mr Folair                           Annual  2  29/9/89   11
    Mr Gregsbury                        NOT PAID             12
    Arthur Gride                        Annual  2  4/12/89   13
    Mrs Grudden                201 3726 NOT PAID             15
    Sir Mulberry Hawk                   Annual  4  4/12/89
    Mr and Mrs Kenwig

                                        Annual  2  5/10/89   16
                                        Annual  2  20/1/90   17
DAT2                                    Annual  2  10/1/90   18
        Miss Knag                       Annual  2            20
        Miss La Creevy                  NOT PAID             21
        Miss Ledbrook                   NOT PAID             22
        Mr and Mrs Lenville             Life                 24
        Mr Lillyvick                    NOT PAID
        Tim Linkinwater
        Mr and Mrs Mantalini
```

A Treasurer's Companion



Beebug's
lower-case
Helvetica
for SPLINE2

OutFont

ALL THIS FOR £4.75 (5.25" & 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) available at the same prices.

DISC (5.25" or 3.5") SUBSCRIPTION RATES	UK ONLY	OVERSEAS
6 months (5 issues)	£25.50	£30.00
12 months (10 issues)	£50.00	£56.00

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

BEEBUG, 117 Hatfield Road, St.Albans, Herts AL1 4JS

Special Offers to BEEBUG Members March 1991

BEEBUG'S OWN SOFTWARE

Code	Product	Members Price inc.VAT
1407a	ASTAAD3 - 5" Disc (DFS)	5.95
1408a	ASTAAD3 - 3.5" Disc (ADFS)	5.95
1404a	Beebug Applics I - 5" Disc	4.00
1409a	Beebug Applics I - 3.5"Disc	4.00
1411a	Beebug Applics II - 5" Disc	4.00
1412a	Beebug Applics II - 3.5" Disc	4.00
1421b	Beebug Binder	4.20
1600a	Beebug magazine disc	4.75
1405a	Beebug Utilities - 5" Disc	4.00
1413a	Beebug Utilities - 3.5" Disc	4.00
1450a	EdiKit 40/80 Track	5.75
1451a	EdiKit EPROM	7.75
1452a	EdiKit 3.5"	5.75
0005b	Magscan Vol.1 - 8 40 Track	12.50
0006b	Magscan Vol.1 - 8 80 Track	12.50
0011a	Magscan Update 40 track	4.75
0010a	Magscan Update 80 track	4.75

Code	Product	Members Price inc.VAT
PAG1	Arcade Games (5.25" 40/80T)	5.95
PAG2	Arcade Games (3.5")	5.95
PBG1	Board Games (5.25" 40/80T)	5.95
PBG2	Board Games (3.5")	5.95
0077b	C - Stand Alone Generator	14.25
0081c	Masterfile ADFS M128 80 T	16.50
0024c	Masterfile DFS 40 T	16.50
0025c	Masterfile DFS 80 T	16.50
0085c	Printwise 40 Track	22.50
0086c	Printwise 80 Track	22.50
0009c	Studio 8	16.50
0074c	Beebug C 40 Track	44.25
0075c	Beebug C 80 Track	44.25
0084c	Command	29.25
0073c	Command(Hayes compatible)	29.25
0053c	Dumpmaster II	23.25
0004c	Exmon II	24.00
0087c	Master ROM	29.25

*Please add p&p. **UK** : **a** - 60p, **b** - £1.50, **c** - £2.50 , **Europe: a** - £1.00, **b** - £1.50, **c** - £9.30*

OTHER MEMBERS OFFERS

Due to a popular demand we are extending the offers from last month. The products listed below will be available for a limited period only while stocks are available.

LISP

RRP £20.00 (inc VAT)

Offer price £2.99 (inc VAT) + £0.60 p&p

The fundamental LISP language on ROM (no manual).

Stock code 1003a

FORTH

RRP £20.00 (inc VAT)

Offer price £2.99 (inc VAT) + £0.60 p&p

A complete implementation of the FORTH language on ROM (no manual).

Stock code 1041a

ViewSheet

Normal Members price £42.49 (inc VAT)

Offer price £12.50 (inc VAT) + £1.50 p&p

Acorn's excellent spreadsheet for the BBC micro and Master, supplied on a 16K ROM. A very powerful and extremely useable product which will produce results ready to print or for direct merging into View text files.

Stock code 1001b

ViewSpell

Normal Members price £28.75 (inc VAT)

Offer price £7.00 (inc VAT) + £1.50 p&p

The automatic spelling checker with a built-in 75 000 word dictionary. Supplied on ROM with full manual, examples disc and reference card. Ideal for View or ASCII text files, and can use updateable user dictionaries.

Stock code 1043b

ViewStore

Normal Members price £42.49 (inc VAT)

Offer price £13.80 (inc VAT) + £1.50 p&p

ViewStore is Acorn's database manager program. It offers 40 and 80 column display, multiple indexes, detailed report generation, variable field lengths and much, much more. Excellent value for money.

Stock code 1019b