Vol.12 No.9 March 1994

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

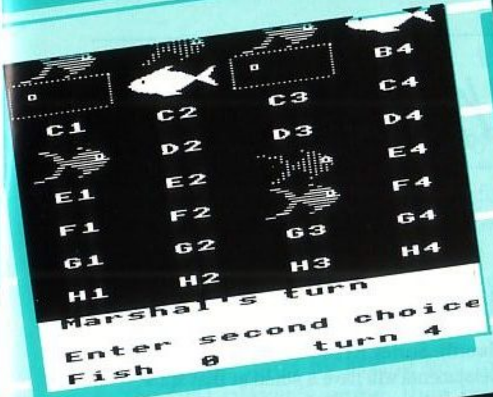
- EXTENDED KEYBOARD
- BODY BUILDING
- EARTHWARP
- EXTENDED GRAPHICS WITH LOGOTRON LOGO

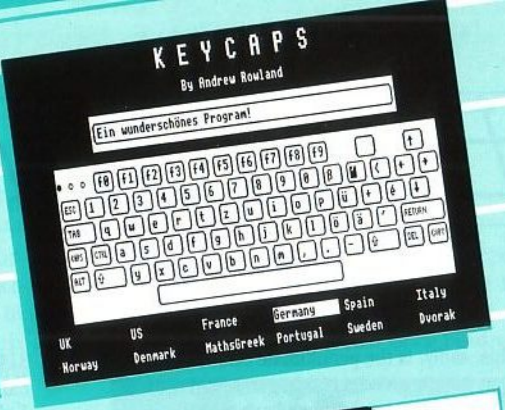
BEEBUG

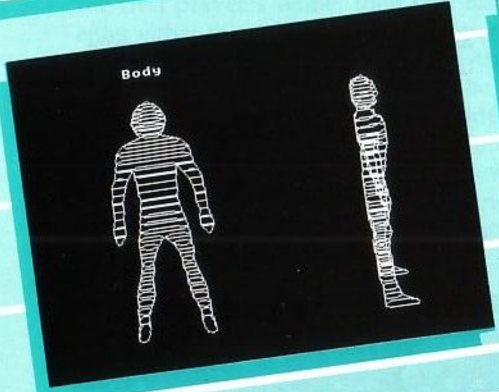
Vol.12 No.9 March 1994

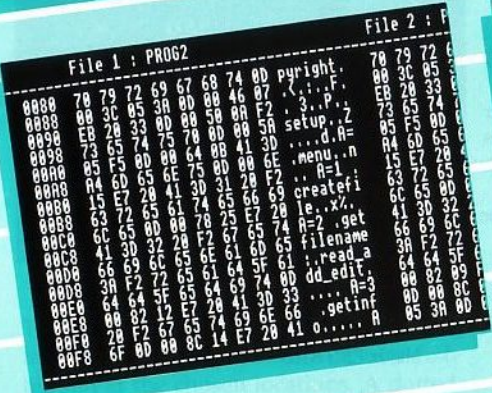
BEEBUG MAGAZINE is produced by RISC Developments Limited.

Printed by Arlon Printers (0923) 258328 ISSN-0253-7561

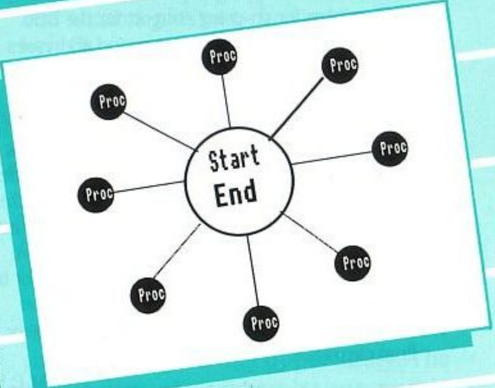








```
                                        File 2 : F
      File 1 : PROG2
---------------------------------------------------
0080   70 79 72 69 67 68 74 0D pyright.   70 79 72
0088   00 3C 05 3A 0D 00 46 07 .<.:..F.   00 3C 05
0090   EB 20 33 0D 00 50 0A F2 . 3..P..   EB 20 33
0098   73 65 74 75 70 0D 00 5A setup..Z   73 65 74
00A0   05 F5 0D 00 64 0B 41 3D ....d.A=   05 F5 0D
00A8   A4 6D 65 6E 75 0D 00 6E .menu..n   A4 6D 65
00B0   15 E7 20 41 3D 31 20 F2 .. A=1 .   15 E7 20
00B8   63 72 65 61 74 65 66 69 createfi   63 72 65
00C0   6C 65 0D 00 78 25 E7 20 le..x%.    6C 65 0D
00C8   41 3D 32 20 F2 67 65 74 A=2 .get   41 3D 32
00D0   66 69 6C 65 6E 61 6D 65 filename   66 69 6C
00D8   3A F2 72 65 61 64 5F 61 :.read_a   3A F2 72
00E0   64 64 5F 65 64 69 74 0D dd_edit.   64 64 5F
00E8   00 82 12 E7 20 41 3D 33 .... A=3   00 82 09
00F0   20 F2 67 65 74 69 6E 66  .getinf   0D 00 8C
00F8   6F 0D 00 8C 14 E7 20 41 o..... A   05 3A 0D
---------------------------------------------------
```



Editor: Mike Williams
Assistant Editor: Kristina Lucas
Editorial Assistance: Marshal Anderson
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Subscriptions: Helen O'Sullivan
Managing Editor: Sheridan Williams

BEEBUG

117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 840303, FAX: (0727) 860263

Office hours: 9am-5pm Monday to Friday
Showroom hours: 9am-5pm Monday to Saturday
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

Editor's Jottings/News

This, sadly, is the penultimate issue of BEEBUG. Next month we will be saying our final farewells. In that issue I hope to look back over the past twelve years, and highlight some of the more outstanding events for BEEBUG and the BBC micro.

Normally, we produce a printed index for each volume which is sent out with the first issue of the succeeding volume. This time the index will accompany the last issue of volume twelve. We will, if space permits, include a full Magscan index for volume twelve on next month's magazine disc.

The user groups which I highlighted last month have reported a not unexpected upsurge in interest. It is to these groups that we suggest readers turn for on-going support for the BBC micro. For reference their names and addresses are repeated at the foot of this page.

You will find enclosed with this issue of BEEBUG a leaflet giving details of our final offers on back issues of BEEBUG and magazine discs. This offer will be repeated next month. Please note that this is your last opportunity to fill in any gaps you may have. It does mean, however, that once we have run out of stock on any item, then we shall not be producing any more. Our associated *Best of BEEBUG* range is also being reduced in price for a final clearance.

M.C.W.

SHOWS AND EVENTS

The second **April Acorn User Show** will be held at the Harrogate International Centre from 22nd - 24th April. Doors will be open 10.00am to 5.00pm daily. Advance tickets are £4.00 (£5.00 at door), under 16 £2.00 (£2.50), family £10.00 (£13.00) - 2 adults + 2 children. Telephone 0734 814713 for ticket sales, or contact Safesell Exhibitions Ltd., Market House, Cross Road, Tadworth, Surrey KT20 5SR. BEEBUG/RISC Developments will have a stand at that show.

All Formats Computer Fairs continue apace. Latest dates and venues are as follows:

Mar 6	Brunel Centre, Temple Meads, Bristol.
Mar 13	Tolworth Recreation Centre, Surbiton (A3).
Mar 19	Haydock Park Racecourse (J23 M6).
Mar 20	National Motorcycle Museum, NEC, Birmingham (J6 M42).
Mar 26	Washington Leisure Centre, Washington Dist. 1.
Mar 27	Woodside Hall, St George's Cross, Glasgow.

USER GROUPS

Beeb Developments User Group,
73 Spital Crescent, Newbiggin-by-Sea, Northumberland NE64 6SQ, tel. 0670 521055.

8 Bit Software,
17 Lambert Park Road, Hedon, Hull HU12 8HF, tel. 0482 896868.

ByteBack,
33 King Henry Mews, Enfield Lock, Middx EN3 6JS.

Solinet,
41 Wentworth Drive, Rainworth, Mansfield, Nottingham NG21 0FB.

Destroyed Realities Disc Based Magazine,
82 Main Street, Pembroke, Dyfed, Wales SA71 4HH.

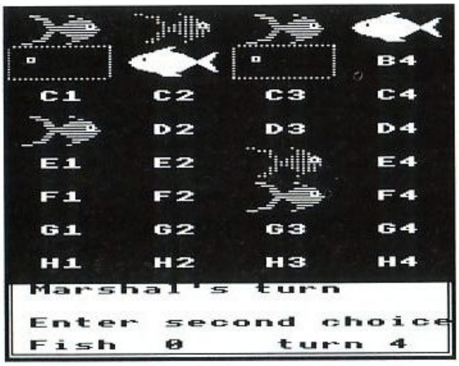
Please enclose an SAE when writing to any of these user groups.

Fish

Have a whale of a time with Alan Gray's game.

Fish is a memory testing game for two players, or one player against the computer. The object is to try to match up pairs of identical fish which are hidden at 32 labelled screen positions.

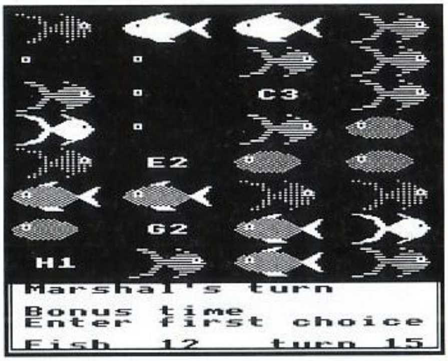
The game starts by asking you to enter the first player's name, then you may enter either the second player's name or press Return to play against the computer. In either case there will be a short delay while the 32 fish are hidden in a random order at screen positions labelled from A1 to A4 on the top row, down to H1 to H4 on the bottom row. Note that you will need to have the Caps Lock on for all these inputs.



```
Marshal's turn
Enter second choice
Fish 0      turn 4
```

Figure 1. Near the start of a game

The first player enters any two of these labels, thereby revealing the two fish hidden at the chosen locations. A dotted rectangle is temporarily drawn round the current selections, which helps when the screen is displaying a lot of previously 'caught' fish. If the fish happen to be identical, they remain on screen and are added to the first player's catch. The first player continues entering pairs of labels until they fail to make a match. The unmatched fish are then replaced by their position labels which are then coloured yellow, to indicate that they have been chosen at least once before. . . .

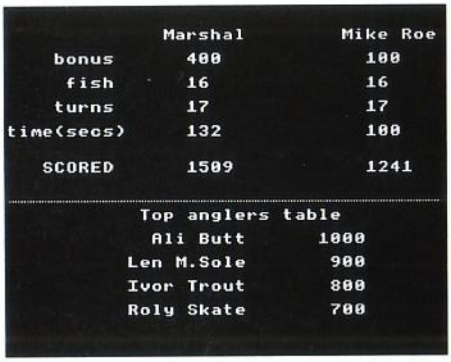


```
Marshal's turn
Bonus time
Enter first choice
Fish 12     turn 15
```

Figure 2. Game nearly over

The second player is invited to choose two fish, and continues with the game until they fail to make a match. The first player then takes over, with play passing back and forth until all the fish are paired up. There are 8 different types of fish (2 pairs of each type), and during play it is possible to earn bonus points by matching any of these pairs when 'BONUS TIME' appears in the text window at the bottom of the screen; a distinctive sound is also used to draw your attention to this opportunity. The possibility of earning bonus points occurs at random intervals, so if you have a good memory, and a little luck, you will be able to increase your lead over your opponent.

At the conclusion of the game, the scores are calculated, based on the number of fish caught divided by the number of guesses you took, plus any bonus points earned. To avoid a drawn game, the amount of thinking time used when choosing fish is also taken into account. Your score is recorded on the 'Anglers Table' if it is higher than one of the current scores. It would be a nice idea to add a save routine to these scores.......



```
              Marshal           Mike Roe
   bonus      400               100
    fish      16                16
   turns      17                17
time(secs)    132               100

  SCORED      1509              1241

          Top anglers table
             Ali Butt        1000
           Len M.Sole         900
           Ivor Trout         800
           Roly Skate         700
```

Figure 3. The scoring

PROGRAMMING NOTES

A call of *FX220,0 disables the Escape key, to avoid accidently pressing Escape when entering the number one during play. You can escape by pressing Ctrl and @ together. Mode 5 is used in order to free sufficient memory to create the 58 characters needed in the design of the fish. There are two basic shapes for the fish, each created from 12 user defined characters, but the striped and spotted effects require an additional 24 characters. The default number of user defined characters is 32, so the character set needs to be expanded by issuing a call of *FX20,1. This enables 32 more characters to be defined, but also requires an extra page of memory.

GETTING SPOTTED

The striped and spotted effects on the fish are created by using undocumented GCOL action numbers, e.g. GCOL 128,1 gives vertical stripes. The characters are defined at lines 1100-1670, then assembled into 5 intermediate shapes at lines 1690-1730, and formed into the 8 different varieties at lines 1740-1810. Pre-assembling these final shapes into an array uses more memory but dramatically reduces the time taken to display the fish shapes.

The trickiest part of the program was to make the computer play at a level at which it could be beaten, but not too easily. This is achieved by allowing it to remember the positions of the previous six guesses, and checking these for matches in *PROCtmf*. If there are no matches, it chooses a position at random, and re-checks its memory. The computer is given a penalty of 50 seconds, so it doesn't have an unfair thinking time advantage. If you find that you can't beat the computer, try either deleting lines 2640 to 2670 or change them to REM statements.

```
   10 REM Program Fish
   20 REM Version B1.0
   30 REM Author  Alan Gray
   40 REM BEEBUG  March 1994
   50 REM Program subject to copyright
   60 :
  100 *FX20,1
  110 *FX220,0
  120 PROCinit
  130 REPEAT
  140 MODE4
  150 VDU19,1,3;0;19,0,4;0;
  160 PROCnames:PROCreset
  170 py%=1
  180 MODE5
  190 VDU19,0,4;0;
  200 PROClabels:VDU4,12,17,0
  210 PRINTTAB(1,0)pl$(py%);"'s turn"
  220 REPEAT:TIME=0
  230 PROCchoose
  240 time%(py%)=time%(py%)+TIME
  250 turn(py%)=turn(py%)+1
  260 IF ch(1)=ch(2) SOUND1,-15,200,10 E
LSE SOUND 1,-15,16,10
  270 z=INKEY(400)
  280 IF ch(1)<>ch(2) PROCrub ELSE PROCh
old
  290 bt%=0:VDU19,3,7;0;
  300 PROCscore
  310 UNTIL n%=16
  320 z=INKEY(200):VDU4,12
  330 PRINT'" PRESS SPACE BAR":z=GET
  340 MODE4:PROCtable
  350 PRINTTAB(2,29)"Another game? Y/N"
  360 *FX15,0
  370 q$=GET$:UNTIL q$="N"
  380 END
  390 :
 1000 DEFPROCinit
 1010 ENVELOPE 1,2,-2,1,63,37,75,50,63,0
,0,-1,63,127
 1020 ENVELOPE 2,7,132,124,7,30,30,50,12
7,0,0,-2,126,126
 1030 DIM xp%(2),yp%(2),ch(2),bonus%(2)
 1040 DIM fco(2),turn(2),time%(2),bl(24)
 1050 DIMpts%(2),tab%(5),tab$(5),pl$(2),
n$(2)
 1060 DIM cm%(6),fish%(32),tally%(32)
 1070 DIM fs$(8),f$(5)
 1080 g$=CHR$18
 1090 bk$=CHR$11+CHR$8+CHR$8+CHR$8
 1100 VDU23,128,0,0,0,0,0,1,7,31
 1110 VDU23,129,0,3,15,62,127,255,255,25
5
 1120 VDU23,130,0,128,0,0,0,192,240,248
 1130 VDU23,131,0,0,0,0,0,3,6,14
 1140 VDU23,132,63,127,127,255,255,127,1
27,63
 1150 VDU23,133,255,255,255,255,255,255,
255,255
 1160 VDU23,134,252,254,255,255,255,255,
254,252
 1170 VDU23,135,30,60,124,248,248,124,60
,30
 1180 VDU23,136,31,7,1,0,0,0,0,0
 1190 VDU23,137,255,255,255,63,31,15,0,0
 1200 VDU23,138,248,240,224,128,192,224,
0,0
 1210 VDU23,139,14,6,3,0,0,0,0,0
 1220 VDU23,140,0,0,0,0,0,0,1,7,0
 1230 VDU23,141,0,3,15,0,0,255,255,0
 1240 VDU23,142,0,128,0,0,0,192,240,0
 1250 VDU23,143,0,0,0,0,0,3,6,0
 1260 VDU23,144,0,127,127,0,0,127,127,0
 1270 VDU23,145,0,255,255,0,0,255,255,0
 1280 VDU23,146,0,254,255,0,0,255,254,0
 1290 VDU23,147,0,60,124,0,0,124,60,0
 1300 VDU23,148,0,7,1,0,0,0,0,0
 1310 VDU23,149,0,255,255,0,0,15,0,0
 1320 VDU23,150,0,240,224,0,0,224,0,0
 1330 VDU23,151,0,6,3,0,0,0,0,0
 1340 VDU23,152,0,0,0,0,0,0,5,10
 1350 VDU23,153,0,0,0,0,85,170,85,170
 1360 VDU23,154,0,0,0,0,0,128,80,168
 1370 VDU23,155,0,0,0,0,0,0,0,0
 1380 VDU23,156,21,42,85,170,85,42,85,42
 1390 VDU23,157,85,170,85,170,85,170,85,
170
 1400 VDU23,158,84,170,85,170,85,170,84,
168
 1410 VDU23,159,21,2,1,0,0,0,0,0
 1420 VDU23,160,85,170,85,0,0,0,0,0
 1430 VDU23,161,80,160,64,0,0,0,0,0
 1440 VDU23,162,0,60,7,3,1,0,0,0
 1450 VDU23,163,0,0,128,224,240,112,56,5
6
 1460 VDU23,164,0,48,24,14,3,15,31,63
 1470 VDU23,165,0,0,0,0,0,192,224,240
 1480 VDU23,166,60,28,31,15,15,31,28,60
 1490 VDU23,167,127,255,255,255,255,255,
255,127
 1500 VDU23,168,240,248,248,252,252,248,
248,240
 1510 VDU23,169,0,0,0,1,3,7,60,0
 1520 VDU23,170,56,56,112,240,224,128,0,
0
 1530 VDU23,171,63,31,15,3,14,24,48,0
 1540 VDU23,172,240,224,192,0,0,0,0,0
 1550 VDU23,173,0,60,0,3,0,0,0,0
 1560 VDU23,174,0,0,0,224,0,112,0,56
 1570 VDU23,175,0,48,0,14,0,15,0,63
 1580 VDU23,176,0,0,0,0,0,192,0,240
 1590 VDU23,177,0,28,0,15,0,31,0,60
 1600 VDU23,178,0,255,0,255,0,255,0,127
 1610 VDU23,179,0,248,0,252,0,248,0,240
 1620 VDU23,180,0,0,0,1,0,7,0,0
 1630 VDU23,181,0,56,0,240,0,128,0,0
 1640 VDU23,182,0,31,0,3,0,24,0,0
 1650 VDU23,183,0,224,0,0,0,0,0,0
 1660 VDU23,184,7,7,7,7,0,0,0,0
 1670 VDU23,185,0,2,2,0,0,0,0,0
 1680 n$(1)="first":n$(2)="second"
 1690 FOR k%=0 TO 5:f$(k%)=CHR$11+CHR$8
 1700 FOR j%=1 TO 22
 1710 READ bl(j%)
 1720 f$(k%)=f$(k%)+CHR$(bl(j%))
 1730 NEXT:NEXT
 1740 FOR j%=1 TO 8
 1750 READ f1%,c1%,c2%,ol%
 1760 fs$(j%)=g$+CHR$(c1%)+CHR$(c2%)+f$(
f1%)+bk$
 1770 IF ol% READ f1%,c1%,c2%
 1780 IF ol% fs$(j%)=fs$(j%)+g$+CHR$(c1%
)+CHR$(c2%)+f$(f1%)
 1790 NEXT
 1800 eye$="":FOR e%=1TO9:READ cs%
 1810 eye$=eye$+CHR$(cs%):NEXT
 1820 FOR j%=1 TO 4:READ tab$(j%),tab%(j
%):NEXT
 1830 ENDPROC
 1840 DEFPROCnames
 1850 FOR py%=1 TO 2:CLS
 1860 PRINTTAB(10,5);"Fishing Match"
 1870 PRINTTAB(2,12);"ENTER the name of
";
 1880 PRINT n$(py%);" player"'
 1890 IF py%=2 PRINTTAB(2,17);"or Press
return to play computer"
 1900 INPUT pl$(py%):NEXT
 1910 IF pl$(2)="" THEN pl$(2)="Mike Roe
"
 1920 ENDPROC
 1930 DEFPROCreset:CLS
 1940 FOR j%=1 TO 32
 1950 tally%(j%)=0:fish%(j%)=0
 1960 NEXT
 1970 FOR j%=1TO6:cm%(j%)=0:NEXT
 1980 FOR j%=1TO2
 1990 bonus%(j%)=0:fco(j%)=0
 2000 time%(j%)=0:turn(j%)=1:NEXT
 2010 PRINTTAB(2,10);"There will be a sh
ort delay,while"
 2020 PRINT'"  4 of each variety of 8 fi
sh types"
 2030 PRINT'"  are randomly hidden at 32
 screen"
 2040 PRINT'"  positions, labelled A1 to
 H4."
 2050 FOR F%=1 TO 32
 2060 REPEAT
 2070 V%=RND(8):T%=0
 2080 FOR C%=1 TO F%
 2090 IF V%=fish%(C%) T%=T%+1
 2100 NEXT
 2110 UNTIL T%<4
 2120 fish%(F%)=V%
 2130 NEXT
 2140 ENDPROC
 2150 DEFPROClabels
 2160 VDU12,5
 2170 FOR x%=1 TO 4
 2180 FOR y%=1 TO 8
 2190 MOVE x%*320-224,1064-y%*96
 2200 a$=CHR$(y%+64)+CHR$(x%+48)
 2210 PRINT a$
 2220 NEXT:NEXT
 2230 TIME=0:n%=0
 2240 VDU28,0,31,19,25,4,17,131,12
 2250 ENDPROC
 2260 DEFPROCchoose
 2270 VDU4:PRINTTAB(1,2)"
 "
 2280 bt%=RND(6)
 2290 IF bt%=6 PRINTTAB(1,2)"Bonus time"
:SOUND1,1,120,3
 2300 IF py%=2 AND pl$(2)="Mike Roe" THE
N PROCcomp ELSE PROCman
 2310 ENDPROC
 2320 DEFPROCprfish
 2330 tally%(X%)=1
 2340 xp%(pos%)=x%:yp%(pos%)=y%
 2350 VDU5
 2360 GCOL0,0
 2370 MOVE fx%,fy%:PRINT f$(0)
```

Continued on page 48

Extended Keyboard (Part 1)

A customised keyboard and much more from Andrew Rowland.

Because of their length, the programs described here are available only on this month's magazine disc.

If you ever need to type some of the less usual characters like 'é' in café, use Maths and scientific symbols like $^1/_2$ or µ or type in a foreign language, you may have been frustrated - unless you have a Master Compact - at the lack of facilities the Beeb has for such things. It compares badly with PCs and the Archimedes in this respect.

This program gives you all the features of a PC's keyboard and more: 'one touch' foreign and special characters, using the Shift Lock key as a special shift key, effectively providing a whole alternative keyboard. It also allows you to redefine the keyboard and alter the effect of nearly every key, including Shift and Ctrl combinations, giving you a huge number of characters easily available from the keyboard. You can swap between different keyboard layouts at the touch of a button and a number of standard international layouts are supplied. Although the utility will work with *any* program, it is particularly effective with View. It also provides a keypress to print a text screen - a useful and popular feature of PCs.

To use this package, you will need a Master 128 with the original operating system (3.20 - type *FX0 to check) and preferably a printer which incorporates the IBM character set. If your printer is different, some adaptations will be required, these will be explained next month.

GETTING GOING

Copy the following files from the monthly disc onto a blank disc or empty ADFS directory:

IBMdata
AltRomB
KeyMapG
PDgen
KeyCaps
ReadMe
Patch

First run *KeyMapG* which creates two data files, *Countrs* and *KbdData*. Then run *AltRomB* which assembles *AltRom*, the ROM image. Finally run *PDgen* which creates the View printer driver. The ReadMe file contains further information, including the use of *Patch*.

Now you should load AltRom into sideways RAM using:

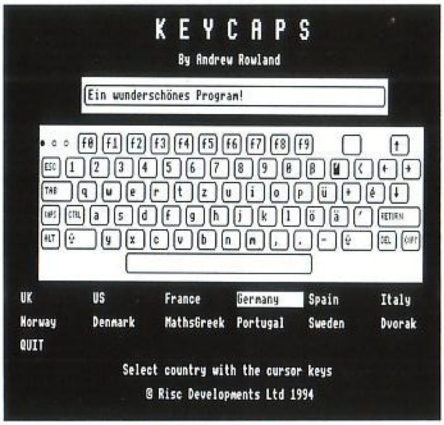
```
*SRLOAD AltRom 8000 <socket>
```

Then press Ctrl-Break to initialise it, or use the Master ROM's *INIT and just press Break. You will find the Shift Lock key goes out, if it was on, and the Shift Lock key becomes a special shift key which produces foreign and other special characters in conjunction with other keypresses. It makes available all the characters in your printer with ASCII codes over 127, including some useful ones for drawing lines and boxes for tables etc. I will call all these 'special characters' from now on.

GUIDED TOUR

The best way to get to know what features are available is to run KeyCaps.

This provides a visual representation of the keyboard so you can see at a glance which characters are available.


KEYCAPS
By Andrew Rowland

Ein wunderschönes Program!

Select country with the cursor keys
© Risc Developments Ltd 1994

The KeyCaps screen

The Shift-Lock key becomes an Alt key. Hold it down and try pressing another key with or without Shift. Caps Lock will affect which character you get (try Alt-A and Alt-a to get lower and capital Ä, for example).

Experiment holding in turn Shift, Ctrl, Alt (as we shall call the Shift Lock key from now on), pressing Caps Lock, and using them in combination. The display will change to reflect the characters available. Control codes (i.e. those less than ASCII 32) are shown in white on black. You can type too, and the results will appear in a small window. Any blank keys in the display do not produce letters when pressed. Try not to type when the display is being updated, as KeyCaps misses these keystrokes (this is not a problem outside KeyCaps though).

At the bottom of the screen is a menu of the countries available. Select a different one by holding a cursor key down until the highlight moves to the required country and release. Selecting *Quit* in this way leaves the program with the last country still selected. There is a quicker way to choose any of the first nine countries: press Alt (wait for the display to update) and press f0 to f8.

Each 'country' is in fact a different keyboard layout to match the typewriters of the particular country, and is intended for people who learned to touch type on a foreign typewriter or who need the letters of another language easily available. Anyone who also uses another type of computer may like the US layout, which matches most other keyboards. Two of the options are not in fact countries: Dvorak is an alternative to QWERTY which is supposed to be a lot more efficient and to speed up touch typing, though such is the prevalence of QWERTY that it never caught on. *MathsGreek* is the same as the UK but if you press Alt a lot of extra mathematical and scientific characters are available, including all the Greek letters your printer has. With the exception of MathsGreek, the Alt key produces the same letters no matter which country is selected and I have chosen them to be memorable and useful rather than numerous. This, and indeed any of the layouts, can be changed with complete freedom and I will tell you how to do this next month.

Some countries have special accent keys which can be distinguished from others by an accent shown white on black - the ^ key in the German layout is an example. Press one of these and the cursor becomes large to show it is waiting for a second keypress. If the next keypress is a vowel, an accented foreign character will appear; if a space, / (or \," or ^) will appear. If neither, it peeps a warning and accepts the second keypress.

When you have selected your preferred layout and familiarised yourself with it, choose Quit from the menu without stopping at any other country on the way. All your settings remain in force (another way of setting up the system is described below). There are two more features which are not available within KeyCaps which you can try now.

If you know the ASCII code of the character you require, hold down Alt and type the number in decimal on the numeric keypad and release the Alt key. The character will appear as if it had been typed in normally.

Pressing Alt f9 or Shift-Alt f9 whenever the computer is waiting for input will cause any text on the whole screen to be printed in fast text mode: this is our version of PCs' 'Print Screen' or 'PrtSc' key. Graphics or unrecognised characters will be replaced with a space, and it can be cancelled any time by pressing Escape. This is a great way of getting information off the screen, even if the program you are using doesn't have a print option, and without having to wait for a slow graphics dump.

Typing Shift-Alt f9 prints all characters, including foreign ones. Use Alt f9 if you want it to ignore foreign and special characters because, for instance, you are not using the IBM character set. It is the one to use in Edit's K or D mode, for instance. PrtSc can cope with text in reverse video and will be reasonably intelligent with mode 7 screens, but it cannot reproduce mode 7 graphics.

ALTROM IN USE

Although intended to work with View, AltRom will work with almost any program, but you need to be aware of what is happening. The Master uses codes above 127 to represent function and cursor keys. Because of this, if you use AltRom in, say, Edit, you may find unexpected effects when trying to enter a special character - it mimics a function key instead. The thing to do is to experiment and see which letters are OK and which to avoid. Special characters will, however, never be expanded into strings you have set up using *KEY. If you type special characters when entering lines of Basic, they will have been turned into keywords when you list the program (unless between speech marks) - a useful shortcut.

Since the ROM changes the Master's character set into one which is compatible with your printer it may spoil the screen display of some programs (e.g. Edit's K or D mode). The old character set may be reinstated with *MASTER, the new one selected with *IBM. It also follows that the ROM is of little use in mode 7 since the new characters can't be displayed in this mode.

ALTROM AND VIEW

None of the above limitations (apart from the use of mode 7) apply when using View. Special characters are inserted painlessly into the text and treated just like any others. This is truly WYSIWYG - no messing round with odd highlight sequences. You do need the new printer driver, *Epson*. If your printer is not Epson compatible (most are) the data statements in PDgen can be altered to tailor the driver to your printer.

This driver mimics exactly the Acornsoft Epson printer driver, supporting all the functions described in the View manual or on the Quick Reference card which

Continued on page 54

Extended Graphics with Logotron Logo

Charlie Allingham shows you how on the Master 128.

During the summer holidays I managed to find the time to sit down with my Master and the Logotron Logo manual to refresh my knowledge of the Logo functions that I rarely use. In common, I suspect, with most users I have long ignored many of the facilities provided by the language in order to concentrate on Turtle graphics. What I discovered enhances and builds upon the graphics side of Logo and, perhaps more importantly, provides a natural progression from using relative to absolute co-ordinates.

THE THEORY

As you may know, the BBC's MOS contains a multi-purpose graphics plotting command, PLOT, which is accessible through Basic. You may also know that this is accessible from within Logo by issuing a .SETNIB command, followed by a number between 0 and 255. The default setting is .SETNIB 5, which draws straight lines, and this is automatically selected when Logo is first entered, or after PEN UP and PEN DOWN commands. The Logo manual details other effects available on the BBC 'B', such as dotted lines or the plotting of a single dot, but does not seem to have been updated for the Master series. It therefore fails to explain that the .SETNIB command also gives access to the advanced graphics capability of the BBC Master, in a similar manner to issuing a VDU 25 command from Basic. This enables the plotting of filled, and sometimes outline, triangles, rectangles, parallelograms, circles, arcs, chord segments, sectors, ellipses and the flood filling of shapes with foreground colours. A list of the most useful numbers is given at the end of this article with details of the number of points required to plot them. Fuller details of the commands are provided in the Master Reference Manual, part one, pages E.3-21 to E.3-34.

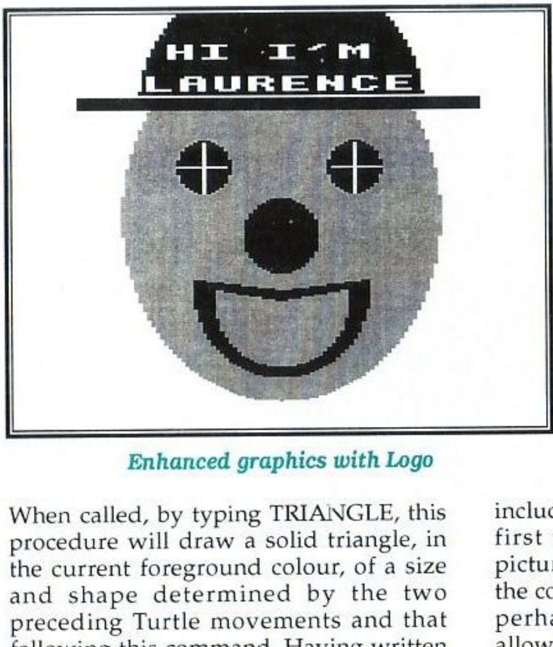
THE PRACTICE

By using these commands in a suitable graphics mode it is fairly quick and easy to make a colourful picture by overlaying different coloured shapes, and with the simplicity of Logo commands the program can be very short. When Logo is used for geometric and recursive patterns it is usual to move the Turtle (cursor) relative to its previous position (e.g. RIGHT 90 FORWARD 200) but there is also a facility to move the Turtle absolutely by using the SETPOS command, which makes life easier when plotting (drawing) shapes that require three cursor positions.

Unusually, the X and Y axes cross at the centre screen in Logo, thus making the centre of the screen position 0 0. The first number denotes the X axis; positive numbers moving to the right, negative ones to the left. The second number denotes the Y axis; positive numbers moving up, negative ones down. Therefore SETPOS [-200 200] (the square brackets are essential) moves the Turtle to a position in the top left-hand quarter of the screen, drawing a line (or whatever) if the pen is down.

Not being one who enjoys working surrounded by pieces of paper I find it simpler to write appropriate one word procedures for each .SETNIB command. It's easy to call them when required within a program, rather than trying to remember a series of apparently meaningless numbers. It is simple to write them in the Logo Editor and then save them all in one file; it is obviously important to choose an appropriate and easily remembered name for the procedure, e.g.

```
TO TRIANGLE
.SETNIB 85
END
```



Enhanced graphics with Logo

When called, by typing TRIANGLE, this procedure will draw a solid triangle, in the current foreground colour, of a size and shape determined by the two preceding Turtle movements and that following this command. Having written your procedures save them - I called mine "EFFECTS - and then load them at the start of each session. When your new work is saved, the procedures will automatically be saved with it.

The table at the end of this article shows which of the functions require two Turtle positions and which require three. In each case the .SETNIB command (the full stop is mandatory) should be made before the final cursor movement - if you make it earlier the effects can be spectacular, but not what you planned. All circular, part circular and elliptical shapes are plotted in an anticlockwise direction. It is very important to remember that to ensure you get the quarter of a circle you wish to draw, and not the other three quarters!

After selecting the Logo language in ROM choose a suitable graphics mode using the SETMODE primitive. Mode 2 gives you the choice of sixteen colours (eight steady and eight flashing colours) and provides the best graphics. Logo will not let you change modes once you have procedures in the memory (it triggers the fault report 'Logo not fresh') so this must be done first. Next, load your procedures (e.g. LOAD "EFFECTS) and away you go.

I have included a short program below, and on the magazine disc, which draws a clown's face as an example. It makes no claims to be structured and is ripe for polishing up, but is included in this fashion, exactly as it was first written, to show how quickly a picture can be built up in this way. All the commands should be familiar, except perhaps the VDU 5 command which allows text to be printed in the graphics screen below the Turtle. To load the program from disc first enter Logo and set the mode to 2. Now type LOAD "CLOWN and press Return; you should see that a number of procedures are defined. After that simply typing CLOWN1 and pressing Return will run the procedure and draw the clown's face.

5	Draws solid lines (*2 points*).
21	Draws dotted lines (*2 points*).
69	Draws a single pixel(*2 points*).
85	Draws a filled triangle (*3 points; the corners*).
101	Draws a filled rectangle. (*2 points; opposite corners*).
117	Draws a filled parallelogram. (*3 points; corners - the fourth being calculated by computer*).
133	Flood fills to non-background. (*2 points*)
149	Draws an outline circle. (*2 points; centre & boundary*).
157	Draws a filled circle as above.
165	Draws a circular arc. (*3 points; centre, first endpoint, second endpoint*)
173	Draws a filled chord segment using the above criteria.
181	Draws a filled sector using .SETNIB 165's criteria.
197	Draws an ellipse outline. (*3 points; the centre, X intercept & high/low point*).
205	Draws a solid ellipse as above.

Useful .SETNIB Commands. (.SETNIB n)

The program can, of course, be listed and edited in the Logo Editor by typing EDIT "CLOWN1 and pressing Return or, if you wish to examine all the procedures, by typing EDALL and again pressing Return. Another helpful command is PR POS, which prints out the Turtle's co-ordinates, extremely useful for a subsequent SETPOS command.

Finally, I hope you will find that this facility is a useful extension to Logo, particularly those of you within the educational field where a natural progression from using relative to absolute co-ordinates is so necessary.

The Clown procedures

```
TO CLOWN1 HT SETPC 7 FILLELLIPSE RT 90
FD 300 SETPOS [0 350] PU
SETPOS [141.924 91.924] SETH 0 PD
SETPC 6 FILLCIRC FD 50 PU BK 50 PD
SETPC 0 REPEAT 4 [FD 45 BK 45 RT 90] PU
SETPOS [-141.924 91.924] SETH 0 PD
SETPC 6 FILLCIRC FD 50 PU BK 50 PD
SETPC 0 REPEAT 4 [FD 45 BK 45 RT 90] PU
SETPOS [0 -35] SETPC 1 PD FILLCIRC FD 70
PU BK 175 SETPOS [-160 -120] SECTOR
SETPOS [160 -120] PU SETPOS [0 -160]
SETPOS [-120 -140] PD SETPC 7 SECTOR
SETPOS  [120  -140]  PU  SETPOS  [0  200]
SETPOS [270 200] SETPC 1 SECTOR
SETPOS [-255 200] FILLRECT
SETPOS [-370 225] PU SETPOS [240 200]
FILLRECT SETPOS [375 225] PU
SETPOS [-370 225] SETPC 0 PD
SETPOS [375 225] PU SETPOS [-255 260]
VDU  [5]  PR  "LAURENCE  PU  SETPOS  [-215
320]
VDU [5] PR "HI\ I'M END

TO FILLRECT .SETNIB 101 END

TO SECTOR .SETNIB 181 END

TO FILLCIRC .SETNIB 157 END

TO FILLELLIPSE .SETNIB 205 END
```

The Logotron Logo Chip is available from:

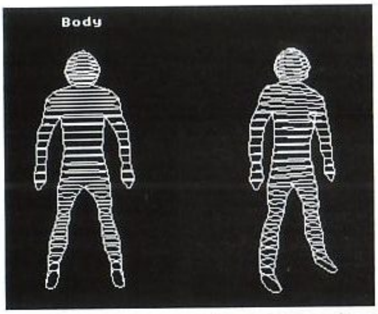
Longman Logotron,
123 Cambridge Science Park,
Milton Road, Cambridge CB4 4ZS.
Tel. 0223 425558.
Price £61.10 inc. VAT.

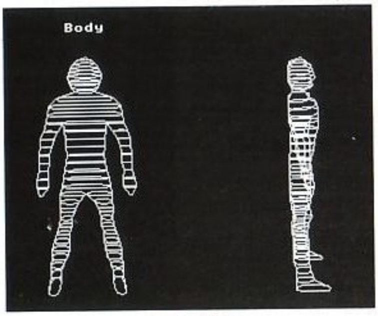
Body Building

by Bill Walker

Several programs have been published over the years for the BBC micro which draw perspective views of three-dimensional objects. In each case, where the object is irregular and cannot be represented by a mathematical formula it has to be defined by a list of co-ordinates of key points on its surface. The 'Spitfire' by I. C. Grant (BEEBUG Vol.3 No.1) was a typical example of this, with over 1200 measurements defining 400 points on the Spitfire's surface.

This program deals with a very different subject and draws a representation of a three-dimensional human body, first from the front, and then from other directions. The program is complete with data (although you can enter your own data as well) so that you can try it straight away. Just type it in and let it run, making sure you have saved a copy first. The program will then display the body as viewed in turn from twelve different directions controlled by the spacebar. Be patient as many calculations are involved.





Body building

The program makes several simplifications which reduce the amount of information that has to be stored. The measurements are so straightforward that you can even replace the values given for those of your own body, and view yourself from different angles!

The first simplification assumes that the body is symmetrical about the middle. This makes the left leg the mirror image of the right, so we only need to measure and store one leg. The same applies for the arms.

The body is then carved up into 'slices'. In the measurements given, the slices are taken at 60mm intervals for the leg, arm and torso, and 30mm intervals (giving more detail) for the head.

Finally, the slices are assumed to have the same shaped outline, which is roughly circular, but to have different lengths and widths. For example, a slice through the foot (a footprint) is long and narrow, a slice through the head is more circular.

This means that the program only stores the position of each slice, and its length and width (five measurements), and from this it can compute the coordinates of 9 points (45 measurements) on the surface. Consequently the number of measurements stored is reduced by a factor of five!

For example:

```
DATA 240,160,0,40,120
```

represents a slice through the foot. Its centre is at X=240, Y=160, Z=0, i.e. 240 mm from the middle of the body, 160mm from the back of the body, and 0mm from the ground. The width of the slice (from centre to side, in the X direction) is 40mm, and the length (from centre to front, in the Y direction) is 120mm (no prizes for working out what size feet I have!).

To take your own measurements, draw round your body to give a front and side outline, and take the measurements shown in figure 1 for each slice. All the measurements should be in millimetres. These measurements replace those in the data statements (the values for the leg start at line 1610, the torso at line 1910, the arm at 2160 and the head (and shoulders) at 2440). The first value in each set is the number of slices that follow. If you take slices at closer intervals than 60mm you will get a more detailed picture, but each 'shot' will take longer to draw.

PROGRAM NOTES

The co-ordinates of points on a slice of width=1, length=1 are computed at the start of the program, and stored in an array. The shape of the slice is a squared circle, which is chosen to give the smoothest join at places where slices with different centres meet, for example where the tops of the two legs meet the bottom of the (one) torso. Values of SINE and COSINE are also computed at the start, to speed up the calculations later.

The main section of the program draws a front view of the body on the left of the screen, and then views it from different angles on the right.

PROCBODY draws the two legs, and notes the leftmost and rightmost points at the tops of the legs. It joins these to the bottom of the torso, then draws the arms. The left and rightmost points of the torso or arms are then joined to the shoulders and head.

PROCe calculates the co-ordinates of the points around the slice, and draws these on the screen, using PROC to map the three-dimensional coordinates to the two-dimensional screen. The left and rightmost points on the slice are recorded for when the slice is joined to the next one.

PROCLEG, PROCARM, PROCTORSO and PROCHEAD each draw bits of the body, joining them up to the other bits where appropriate. See also the many comments in the program for further information.

The techniques described above offer many opportunities for further experiment. You can easily change the overall shape and appearance of the figure by changing the values assigned to the variables ANGLE, VDIST%, ZOOM% and H% in lines 260 to 290. The viewing angle can be changed by modifying the parameters in the main loop from line 370 to 410. Other ideas worth considering might include views from different vertical angles, and even a form of animation using VDU19.

This program was first published in BEEBUG Vol.3 No.9.

```
   10 REM Program Body
   20 REM Version B 1.0
   30 REM Author  B.Walker
   40 REM BEEBUG  March 1994
   50 REM Program subject to copyright
   60 :
  100 MODE4
  110 ON ERROR GOTO 2720
  120 PRINT'SPC(8)"Body"
  130 VDU23,1,0;0;0;0;
  140 REM Build lookup table and ellipse
  150 DIM S(35),C(35),SI(35),CO(35)
  160 FF=.3:REM fiddles elipses,0=no fid
dle
  170 FORI%=0TO35
  180 SI(I%)=SINRAD(I%*10)
  190 CO(I%)=COSRAD(I%*10)
  200 S(I%)=SI(I%)*(1+CO(I%)*CO(I%)*FF)
  210 C(I%)=CO(I%)*(1+SI(I%)*SI(I%)*FF)
  220 NEXT
  230 :
  240 oLXP%=0:oLYP%=0:REM leftmost point
  250 oRXP%=0:oRYP%=0:REM rightmost poin
t
  260 ANGLE=0:REM rotation angle/10
  270 VDIST%=2000:REM viewing distance
  280 ZOOM%=800:REM scale of drawing
  290 H%=1000
  300 :
  310 REM main loop
  320 VDU29,320;512;
  330 PROCBODY(0)
  340 VDU29,0;0;
  350 VDU24,640;0;1279;1023;
  360 VDU29,960;512;
  370 FORA=3 TO 35 STEP3
  380 CLG
  390 PROCBODY(A)
  400 Q=INKEY(500)
  410 NEXT
  420 END
  430 :
 1000 REM Draw Body
 1010 REM ...Draw legs, Draw Torso, Join
 legs with torso
 1020 REM ...Draw Arms, Draw head, Join
arms with head
 1030 DEFPROCBODY(ANGLE)
 1040 PROCLEG(-1)
 1050 TLX%=oLXP%:TLY%=oLYP%
 1060 TRX%=oRXP%:TRY%=oRYP%
 1070 PROCLEG(1)
 1080 IF TLX%<oLXP% THEN oLXP%=TLX%:oLYP
%=TLY%
 1090 IF TRX%>oRXP% THEN oRXP%=TRX%:oRYP
%=TRY%
 1100 PROCTORSO
 1110 TLX%=oLXP%:TLY%=oLYP%
 1120 TRX%=oRXP%:TRY%=oRYP%
 1130 PROCARM(-1)
 1140 IF oLXP%<TLX% THEN TLX%=oLXP%:TLY%
=oLYP%
 1150 IF oRXP%>TRX% THEN TRX%=oRXP%:TRY%
=oRYP%
 1160 PROCARM(1)
 1170 IF TLX%<oLXP% THEN oLXP%=TLX%:oLYP
%=TLY%
 1180 IF TRX%>oRXP% THEN oRXP%=TRX%:oRYP
%=TRY%
 1190 PROCHEAD
 1200 ENDPROC
 1210
 1220 REM Draw ellipse, return lefmost a
nd rightmost points
 1230 DEFPROCe(XOFF%,YOFF%,Z%,A%,B%,LINK
%)
 1240 REM XOFF,YOFF = center of ellipse
 1250 REM ellipse is in plane z=Z
 1260 REM ellipse dia on x axis =A
 1270 REM ellipse dia on y axis =B
 1280 REM LINK=TRUE to join this and las
t ellipse
 1290 LOCAL X%,Y%,K%
 1300 X%=XOFF%:Y%=B%+YOFF%:REM start of
ellipse
 1310 PROCm(X%,Y%,Z%)
 1320 MOVE XP%,YP%
 1330 LXP%=XP%:LYP%=YP%:REM leftmost poi
nt
 1340 RXP%=XP%:RYP%=YP%:REM rightmost po
int
 1350 FORK%=4TO35STEP4
 1360 X%=A%*S(K%)+XOFF%
 1370 Y%=B%*C(K%)+YOFF%
 1380 PROCm(X%,Y%,Z%)
 1390 DRAW XP%,YP%
 1400 IF XP%<LXP%THEN LXP%=XP%:LYP%=YP%
 1410 IF XP%>RXP%THEN RXP%=XP%:RYP%=YP%
 1420 NEXT K%
 1430 X%=XOFF%:Y%=B%+YOFF%
 1440 PROCm(X%,Y%,Z%)
 1450 DRAW XP%,YP%:REM close ellipse
 1460 IF LINK% MOVE oLXP%,oLYP%:DRAW LXP
%,LYP%:MOVE oRXP%,oRYP%:DRAW RXP%,RYP%
 1470 oLXP%=LXP%:oLYP%=LYP%
 1480 oRXP%=RXP%:oRYP%=RYP%
 1490 ENDPROC
 1500
 1510 REM Map 3D coords to 2D screen coo
rds
 1520 DEFPROCm(X%,Y%,Z%)
 1530 LOCAL D,RX%,RY%
 1540 RX%=X%*CO(ANGLE)+Y%*SI(ANGLE)
 1550 RY%=-X%*SI(ANGLE)+Y%*CO(ANGLE)
 1560 D=ZOOM%/(VDIST%-RY%)
 1570 XP%=RX%*D
 1580 YP%=(Z%-H%)*D
 1590 ENDPROC
 1600
 1610 REM Data for one leg
 1620 REM number of ellipses
 1630 DATA12
 1640 REM X,Y,Z of ellipse centre, Axes
of ellipse
 1650 DATA240,160,0,40,120
 1660 DATA235,100,60,38,60
 1670 DATA225,90,120,48,50
 1680 DATA210,80,180,58,55
 1690 DATA200,75,240,63,55
 1700 DATA190,75,300,55,55
 1710 DATA180,75,360,48,50
 1720 DATA160,85,420,50,48
 1730 DATA155,95,480,45,63
 1740 DATA145,80,540,58,65
```

```
1750 DATA130,80,600,70,73
1760 DATA103,80,660,95,80
1770
1780 REM Draw one leg
1790 DEFPROCLEG(SIDE%)
1800 LOCAL I%,C%,X%,Y%,Z%,A%,B%
1810 RESTORE 1630
1820 READC%
1830 READX%,Y%,Z%,A%,B%
1840 PROCe(X%*SIDE%,Y%,Z%,A%,B%,FALSE)
1850 FORI%=2 TO C%
1860 READ X%,Y%,Z%,A%,B%
1870 PROCe(X%*SIDE%,Y%,Z%,A%,B%,TRUE)
1880 NEXT
1890 ENDPROC
1900
1910 REM Data for torso
1920 REM number of ellipses
1930 DATA9
1940 REM X,Y,Z of ellipse centre, Axes
of ellipse
1950 DATA0,85,720,185,83
1960 DATA0,85,780,175,85
1970 DATA0,85,840,160,83
1980 DATA0,90,900,150,85
1990 DATA0,100,960,140,90
2000 DATA0,105,1020,145,95
2010 DATA0,115,1080,150,100
2020 DATA0,130,1140,155,105
2030 DATA0,120,1200,170,113
2040
2050 REM draw torso
2060 DEFPROCTORSO
2070 LOCAL I%,C%,X%,Y%,Z%,A%,B%
2080 RESTORE 1930
2090 READC%
2100 FORI%=1 TO C%
2110 READX%,Y%,Z%,A%,B%
2120 PROCe(X%,Y%,Z%,A%,B%,TRUE)
2130 NEXT
2140 ENDPROC
2150
2160 REM Data for one arm
2170 REM number of ellipses
2180 DATA10
2190 REM X,Y,Z of ellipse centre, Axes
of ellipse
2200 DATA330,145,660,10,5
2210 DATA320,135,720,45,28
2220 DATA320,130,780,40,23
2230 DATA315,120,840,30,25
2240 DATA310,105,900,38,35
```

```
2250 DATA310,100,960,38,40
2260 DATA300,100,1020,40,43
2270 DATA290,90,1080,48,40
2280 DATA265,85,1140,53,55
2290 DATA230,85,1200,60,75
2300
2310 REM Draw one arm
2320 DEFPROCARM(SIDE%)
2330 LOCAL I%,C%,X%,Y%,Z%,A%,B%
2340 RESTORE 2180
2350 READC%
2360 READX%,Y%,Z%,A%,B%
2370 PROCe(X%*SIDE%,Y%,Z%,A%,B%,FALSE)
2380 FORI%=2TOC%
2390 READX%,Y%,Z%,A%,B%
2400 PROCe(X%*SIDE%,Y%,Z%,A%,B%,TRUE)
2410 NEXT
2420 ENDPROC
2430
2440 REM Data for head and shoulders
2450 REM number of ellipses
2460 DATA12
2470 REM X,Y,Z of ellipse centre, Axes
of ellipse
2480 DATA0,115,1260,280,115
2490 DATA0,105,1320,268,105
2500 DATA0,85,1380,220,80
2510 DATA0,75,1440,78,43
2520 DATA0,115,1470,110,78
2530 DATA0,100,1500,123,88
2540 DATA0,105,1530,128,108
2550 DATA0,100,1560,125,100
2560 DATA0,95,1590,115,95
2570 DATA0,90,1620,100,90
2580 DATA0,85,1650,75,70
2590 DATA0,90,1680,30,25
2600
2610 REM Draw one head
2620 DEFPROCHEAD
2630 LOCAL I%,C%,X%,Y%,Z%,A%,B%
2640 RESTORE 2460
2650 READC%
2660 FORI%=1TOC%
2670 READX%,Y%,Z%,A%,B%
2680 PROCe(X%,Y%,Z%,A%,B%,TRUE)
2690 NEXT
2700 ENDPROC
2710
2720 :
2730 ON ERROR OFF:MODE 7
2740 IF ERR=17 END
2750 REPORT:PRINT" at line ";ERL
```

Date Handling Functions and Procedures (Part 1)

by Paul Cuthbertson

Dates occur in a variety of formats, and with the different number of days in each month, can prove a nuisance to handle correctly in programs. This month's Workshop describes a comprehensive set of routines designed to make this task very much easier. These routines will input dates, and check them; they will manipulate dates by adding or subtracting days, whole weeks, months or years; and they will format a date into a string which may take any form from "31/01/94" to "Monday 31 January 1994" and beyond.

All the routines are structured to allow their use within any BBC Basic program. The only global variables are some arrays containing the numbers of days in the month and the names of months and days of the week, and three integers global to some very 'deep' subroutines, which you are unlikely to need on their own.

For those readers who wish to use the software, without necessarily going into the whys and wherefores, simply type in the routines you need, and merge them with your existing program. Just make sure you provide the correct parameters when using the routines.

DATE INPUT/VALIDATION

This routine expects input using a format of DD/MM/YY, DD/MM/YYYY or simply \DD/MM.

If a two digit year is entered then it is assumed to be preceded by '19', i.e. '64' would be stored as 1964. If the year is missed off altogether it is assumed to be 1986 (see line 1080). These features can save a lot of typing for harassed secretaries.

If you enter a date which does not exist for any reason, e.g. 29/2/86 or 31/9/95, the program responds with WHAT?? and requires another input; similarly if you enter any other character apart from a number or a slash.

```
 1000 DEF FNdatein(p$)
 1010 LOCAL date%:date%=FNdateps(FNstrnc
k(p$+" (D/M/.)",3,11,FALSE))
 1020 IF date% THEN=date% ELSE PROCbipa:
=FNdatein("WHAT?")
 1030 :
 1040 DEF FNdateps(date$)
 1050 LOCAL year%,month%,day%
 1060 year%=FNyearis(date$):month%=FNmon
is(date$):day%=FNdayis(date$)
 1070 IF year%<1 =FALSE ELSE IF year%<10
0 year%=year%+1900
 1080 =FNdatepi(day%,month%,year%)
 1090 :
 1100 DEF FNdatepi(d%,m%,y%)
 1110 PROCfeb(y%)
 1120 IF m%<0 OR m%>11 OR y%<1752 OR y%>
9999 OR d%<0 =FALSE
 1130 IF d%>mon%(m%) =FALSE ELSE =d%+m%*
&100+(y%MOD100)*&10000+(y%DIV100)*&10000
```

```
00
 1140 :
 1150 DEF FNlpyr(year%)
 1160 IF year%MOD4<>0 =FALSE ELSE IF yea
r%MOD400=0 =TRUE ELSE IF year%MOD100=0 =
FALSE ELSE=TRUE
 1170 :
 1180 DEF PROCfeb(y%)
 1190 IF FNlpyr(y%) mon%(1)=29 ELSE mon%
(1)=28
 1200 ENDPROC
 1210 :
 1220 DEF FNyearis(date$)
 1230 LOCALa%:a%=INSTR(date$,"/",1)
 1240 IF a%<1 THEN =FALSE ELSE a%=INSTR(
date$,"/",a%+1):IF a%<1 =1986 ELSE =VAL(
RIGHT$(date$,LEN(date$)-a%))
 1250 :
 1260 DEF FNmonis(date$)
 1270 LOCALa%,b%:a%=INSTR(date$,"/",1)
 1280 IF a%<1 =FALSE ELSE b%=INSTR(date$
,"/",a%+1):I Fb% THEN =VAL(MID$(date$,a%
+1,b%-a%-1))-1 ELSE =VAL(RIGHT$(date$,LE
N(date$)-a%))-1
 1290 :
 1300 DEF FNdayis(date$)=VAL(LEFT$(date$
,INSTR(date$,"/",1)))
 1310 :
 1320 DEF PROCbipa:SOUND1,-15,10,10:ENDP
ROC
 1330 :
 1340 DEF FNstrnck(p$,min%,max%,flag%)
 1350 LOCAL b$:@%=1:PRINTp$;:IF flag% GO
SUB 1400
 1360 INPUT"? "b$
 1370 IF LENb$>max% PROCbipa:=FNstrnck("
TOO LONG?",min%,max%,TRUE)
 1380 IF LENb$<min%PROCbipa:=FNstrnck("T
OO SHORT?",min%,max%,TRUE)
 1390 =b$
 1400 PRINT" (MAX "max%;:IF flag% THEN P
RINT" MIN "min%;
 1410 PRINT")";:RETURN
```

TECHNICAL NOTES

The above routine is in the form of a function (FNdatein) which returns an integer representing the date. This integer number is compatible with any other routines requiring the date as a number, and does not need any further processing before being manipulated or formatted. The four bytes of an integer are used as follows: the most significant byte holds the century, the next most significant the parts of a century (i.e. the tens and units of years), the next-to-least significant the months, and the least significant the days. The months and days start at zero rather than one, to allow direct manipulation and referencing of arrays. Within each byte, the number is expressed in binary. Further compression would have been possible, but more awkward than the space saved would justify. The dates may be directly compared, and are easily sorted.

SUMMARY OF FUNCTIONS AND PROCEDURES USED

FNdatein(prompt$) returns an integer containing the date in packed form. This routine is recursive (i.e. calls itself) until a valid date is entered. The parameter supplied is a prompt, with correct punctuation etc added automatically.

FNdateps(date$) converts a date in string form to packed integer. 1900 is added to years between 1 and 99 here.

FNdatepi(day%,month%,year%) converts three integers (day, month and year respectively) into a single packed integer date.

FNlpyr(year%) returns TRUE if the integer passed to it is a leap year, otherwise FALSE. It obeys the 400 year rule.

PROCfeb(year%) accepts the year as an integer parameter, and alters the value of mon%(1) from 28 to 29 if a leap year.

FNyearis(date$) accepts the date as a string, and returns an integer year. The 1986 assumption is made in this routine.
FNmonis(date$) and **FNdayis(date$)** chop the input string and return integers in like manner to FNyearis.

PROCbipa makes a distinctive low tone to signal errors etc.

FNstrnck(prompt$,minlen%,maxlen%,flag%) accepts four parameters and returns a string which has been typed in. More than just a 'mugtrap', it supplies a prompt and other information to the operator. The last parameter is a flag set TRUE to cause display of the length limits as part of the prompt. FNstrnck is recursive until a valid string is input.

THE DATE MANIPULATION SET

These functions and procedures can be used individually for various operations on data held in the standard integer format already described. They are very useful for stepping through diaries or manipulating dates generally. There is a whole set of these, with fairly obvious functions. The full set of routines consists of a day adder, a month adder, a year adder, and a 'days between dates' function.

```
 1500 DEF FNdayip(d%)=d%AND&FF
 1510 DEF FNmonip(d%)=(d%AND&FF00)DIV&10
0
 1520 DEF FNyearip(d%)=((d%AND&FF000000)
DIV&1000000)*100+(d%AND&FF0000)DIV&10000
 1530 :
 1540 DEF FNupday(dt%,nd%)
 1550 LOCALd%,m%,y%:PROCdmyout:d%=d%+nd%
 1560 ON SGN(nd%)+2 GOSUB1570,1600,1610:
PROCyok:=FNdatepi(d%,m%,y%)
 1570 IF d%>0 RETURN
 1580 REPEAT:m%=m%-1:IFm%<0m%=11:y%=y%-1
:PROCfeb(y%)
 1590 d%=d%+mon%(m%):UNTIL d%>0:RETURN
 1600 RETURN
 1610 IF d%<=mon%(m%) RETURN
 1620 REPEAT:d%=d%-mon%(m%):m%=m%+1:IFm%
>11m%=0:y%=y%+1:PROCfeb(y%)
 1630 UNTIL d%<=mon%(m%):RETURN
 1640 :
 1650 DEF FNupmon(dt%,nm%)
 1660 LOCAL d%,m%,y%:PROCdmyout:m%=m%+nm
%
 1670 ON SGN(nm%)+2 GOSUB1680,1700,1710:
PROCyok:=FNdatepi(d%,m%,y%)
 1680 IF m%>-1 RETURN
 1690 REPEAT:m%=m%+12:y%=y%-1:UNTI Lm%>-
1:RETURN
 1700 RETURN
 1710 IF m%<12 RETURN
 1720 REPEAT:m%=m%-12:y%=y%+1:UNTILm%<12
:RETURN
 1730 :
 1740 DEF FNupyear(dt%,ny%)
 1750 LOCAL y%,m%,d%:PROCdmyout:y%=y%+ny
%
 1760 PROCyok:=FNdatepi(d%,m%,y%)
 1770 :
 1780 DEF PROCdmyout
 1790 y%=FNyearip(dt%):m%=FNmonip(dt%)
 1800 d%=FNdayip(dt%):PROCfeb(y%):ENDPRO
C
 1810 :
 1820 DEF PROCyok
 1830 IF y%>9999 y%=9999:d%=31:m%=11:PRO
Cbipa
 1840 IF y%<1752y%=1752:d%=1:m%=0:PROCbi
pa
 1850 PROCfeb(y%):IF d%>mon%(m%)d%=mon%(
m%)
 1860 ENDPROC
```

TECHNICAL NOTES

The first three functions return the day, month or year number as a simple integer from the the date (packed date to integer). FNupday(date%,daystogo%) requires two parameters; the date as a packed integer, and the number of days to be added (this last can be zero or negative). It returns the date as modified, in packed integer form.

FNupmon(date%,monthstogo%) and FNupyear (date%,yearstogo%) do the same as FNupday, for the months and years. When stepping by months from the end of a month, FNupmon will not go beyond a month end; e.g. one month from 31 January gives the last day of February. [To be continued next month.]

This month's Workshop first appeared in BEEBUG Vol.5 No.5.

Public Domain

Alan Blundell rounds up the sources you can contact for PD software.

This month, as promised over the last couple of issues, I've included some information about sources of PD software which has come my way. When I came to write this, I was surprised to learn how few of the PD libraries which have appeared are still operating.

Next issue, I will give details of yet more new software which I have received recently. If space permits, I hope also to look back over the last couple of years to see what has changed on the PD front, and to pick out some of the most successful PD software releases for 8-bit machines.

SOURCES OF PD SOFTWARE

8-Bit Software
17 Lambert Park Road, Hedon, Hull HU12 8HF.

8-Bit Software is certainly still thriving with a small but committed following, and produces regular disc based magazines, which include varying amounts of PD software. Run by Chris J Richardson, the group is primarily a sort of disc-based bulletin board for BBC users, including text articles, software submitted by members and other assorted interesting bits. It also has a PD library of around 150 discs, a large number of which have been contributed to the group by members who have acquired them from the various other PD libraries. Membership of the group is free but there is a charge of 50p per issue of the disc based magazine, and you also need to supply a formatted disc and return postage and packaging for each issue. You don't need to be a member to order PD software; charges are £1.00 per disc, supplied on reused discs, or 50p plus your own disc and return postage and packaging.

BBC PD
18 Carlton Close, Blackrod, Bolton BL6 5DL.

BBC PD has a library of 170 or so PD discs, plus early volumes of BEEBUG magazine discs (up to and including Volume 5), all ELBUG discs, plus most issues of Acorn User's monthly 8-bit magazine discs from 1984 to 1990. It also has 'Fast Access' disc-based magazine, all Disk User magazine discs, and over 30 Megabytes of Master 512 software including the John Lyons range of educational software (as shareware). Charges are £1.50 per 5.25" disc or £1.75 per 3.5" disc, inclusive of VAT, postage and packing.

Mad Rabbit PD
PO Box 4 Crigglestone, Wakefield, West Yorks WF4 3XE.

Mad Rabbit PD is run by Joel Rowbottom and has over the last year brought out several discs which I had not seen previously, especially a selection of discs which are likely to be of interest to 'Star Trek' or 'Red Dwarf' fans. Mad Rabbit reopened, after being closed down for some time, in mid-1993. Unfortunately, I haven't had a response to my last letter and batch of discs, sent in October 1993, so can't give any guarantee that Mad Rabbit is still operational. If it is, charges are £1.25 per disc, fully inclusive. The last catalogue I saw detailed around 50 discs of PD software. (Alan is not the only person to have had problems getting in touch with Mad Rabbit; do write before sending any money to avoid disappointment - Ed.).

I'm afraid that that's the complete list of sources that I have any up-to-date information about. Other addresses which might be worth trying are:

JJF PD
49 Hollyberry Close, Winyates Green, Redditch, Worcester B98 0QT.

Masterdisc
2 Seaview, Hoylake L47 2DD.

I have no information as to the current status of these two, so I don't know if letters will be replied to.

It wouldn't be reasonable to omit SOLINET from a list like this. SOLINET was originally set up as a user group for Solidisk sideways RAM board users, many years ago now (and before the Master series existed). The group still has permission to distribute the full range of software written for these add-ons. Much of this is, however, specific to Solidisk boards rather than to sideways RAM in general, due to the non-standard method used by Solidisk to access RAM on its own expansion boards. However, the group is now of interest to any BBC or Master users who have a disc drive and offers self-help on all sorts of areas of interest from starting to program to building your own hard disc system from scratch (these are examples of topics covered in issues of the group's disc based monthly magazine). The group operates on the basis of a modest annual subscription and members can send a disc monthly for a copy of the latest issue. It also has a PD library available to members.

The contact for SOLINET is:

Ron Marshall, SOLINET, 41 Westbrook Drive, Rainworth, Mansfield, Notts. NG21 0PB.

If you contact any of the above for further information, don't forget to include a reasonably large (A5) stamped, addressed envelope - if not from courtesy, then at least to improve your chances of receiving a reply!

CLOSING DOWN SALE

Unfortunately, having just listed BBC PD (which most readers will know is actually run by myself) as one of the current sources of PD software, I have to announce that it will cease to operate at the end of April 1994. For some time now, I have been struggling to fit in too many demands on my time. Something had to go, and I decided that it had to be BBC PD. I still get lots of correspondence from people interested in PD software, but very few people now want software rather than advice or help with some problem. Rather than let the delays in my replies get longer, I decided to set an end date which will at least give everyone the chance to see what's available.

I hope that readers will forgive me for mentioning this via this column, but I decided that this was the easiest way to circulate the information widely. To give as many people as possible a chance to see what's available, I'm making a limited offer to BEEBUG members (until the end of March) of a free disc and catalogue on receipt of a suitable S.A.E. (at least A5 in size, and preferably reinforced to protect the disc). If you would like one, write to me at the BBC PD address above before the end of March, not forgetting to state which of the following disc formats you prefer: DFS 40-track ; DFS 80-track double-sided ; or ADFS 'L' format. The offer is limited to the number of 'used-once' 5.25" discs which I can make available, but I don't expect to refuse any requests. The discs detail all of the software which I have amassed and include samples of various types of software whenever possible, from games to more serious items.

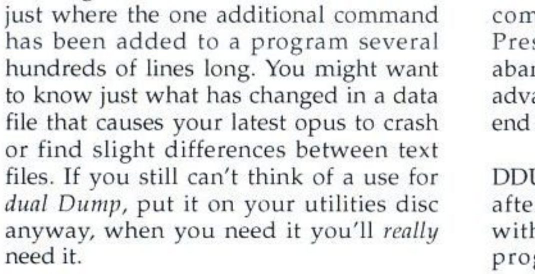
Dual Dump

Compare files with Miroslaw Bobrowski.

At first sight you might wonder just what this program is for. Just how useful is it to be able to simultaneously dump two disc files to the screen? The point is that, when you do need to compare two files, this is the *only* way to do it unless you want to dump files to miles of printout.

You might, for instance, want to find out just where the one additional command has been added to a program several hundreds of lines long. You might want to know just what has changed in a data file that causes your latest opus to crash or find slight differences between text files. If you still can't think of a use for *dual Dump*, put it on your utilities disc anyway, when you need it you'll *really* need it.

Type in the listing below and save it carefully. When run it will assemble the utility and prompt you to press Copy to save the assembled code as *DDUMP*. If you want to save it as anything else then change line 2860.



```
        File 1 : PROG2                               File 2 : PROG3
------------------------------------------------------------------------------------
 0080  70 79 72 69 67 68 74 0D pyright.   70 79 72 69 67 68 74 0D pyright.
 0088  00 3C 05 3A 0D 00 46 07 .<.:..F.   00 3C 05 3A 0D 00 46 07 .<.:..F.
 0090  EB 20 33 0D 00 50 0A F2 . 3..P..   EB 20 33 0D 00 50 0A F2 . 3..P..
 0098  73 65 74 75 70 0D 00 5A setup..Z   73 65 74 75 70 0D 00 5A setup..Z
 00A0  05 F5 0D 00 64 0B 41 3D ....d.A=   05 F5 0D 00 64 0B 41 3D ....d.A=
 00A8  A4 6D 65 6E 75 0D 00 6E .menu..n   A4 6D 65 6E 75 0D 00 6E .menu..n
 00B0  15 E7 20 41 3D 31 20 F2 .. A=1 .   15 E7 20 41 3D 31 20 F2 .. A=1 .
 00B8  63 72 65 61 74 65 66 69 createfi   63 72 65 61 74 65 66 69 createfi
 00C0  6C 65 0D 00 78 25 E7 20 le..x%.    6C 65 0D 00 78 25 E7 20 le..x%.
 00C8  41 3D 32 20 F2 67 65 74 A=2 .get   41 3D 32 20 F2 67 65 74 A=2 .get
 00D0  66 69 6C 65 6E 61 6D 65 filename   66 69 6C 65 6E 61 6D 65 filename
 00D8  3A F2 72 65 61 64 5F 61 :.read_a   3A F2 72 65 61 64 5F 61 :.read_a
 00E0  64 64 5F 65 64 69 74 0D dd_edit.   64 64 5F 65 64 69 74 0D dd_edit.
 00E8  00 82 12 E7 20 41 3D 33 .... A=3   00 82 09 FD 20 41 3E 32 .... A>2
 00F0  20 F2 67 65 74 69 6E 66  .getinf   0D 00 8C 05 E0 0D 00 96 ........
 00F8  6F 0D 00 8C 14 E7 20 41 o..... A   05 3A 0D 00 A0 0B DD 20 .:......
------------------------------------------------------------------------------------
```

Dual Dump in action

To use the program, once it is assembled and saved, just type *DDUMP, making sure the file is on the current disc. The files to be dumped must both be available to the filing system at the same time - on a dual drive machine they can be on separate discs but you must provide full path names.

On being run the program will ask for the two file names and check that they are currently available. If it can't find both files the program stops with an error. The files are then dumped on a mode 3 screen in the same format as the command *DUMP, but side by side. Pressing Escape at any time will abandon the dump, any other key will advance it. The program stops when the end of one of the files is reached.

DDUMP loads and runs from &900, and after it has been loaded it can be run with CALL &900, as long as no other program has used that space in the meantime.

IMPROVEMENTS

There are two ways in which this utility could be improved. Providing a print out in the same format as the dumps appear on screen could be helpful in certain cases. This shouldn't be difficult, doing a Ctrl-B before running DDUMP almost does the trick, it's just a matter of getting rid of the form feeds. It would also be nice to be able to type *DDUMP PROG1 PROG2 and have it happen immediately. A look back at some of Mr Toad's columns should help here.

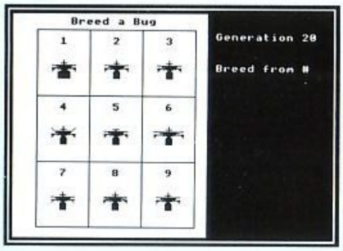
```
   10 REM Program Dual Dump
   20 REM Version B1.00
   30 REM Author  Miroslaw Bobrowski
   40 REM BEEBUG  March 1994
   50 REM Program subject to copyright
   60 :
```

```
 100 offset=&70:buffer=&72
 110 handle=&74:tempy=&75
 120 handle1=&76:handle2=&77
 130 end=&78:end1=&79:end2=&7A
 140 buff1=&7B:buff2=&7D:addr=&80
 150 osfind=&FFCE:osbget=&FFD7
 160 osnewl=&FFE7:oswrch=&FFEE
 170 osword=&FFF1:osbyte=&FFF4
 180 osrdch=&FFE0:osasci=&FFE3
 190 :
 200 FOR pass=0 TO 2 STEP 2
 210 P%=&900
 220 [OPT pass
 230 LDA #0:STA &06
 240 LDA #&40:STA &07
 250 LDA #22:JSR oswrch
 260 LDA #3:JSR oswrch
 270 JSR osnewl
 280 :
 290 LDX #prompt1 MOD &100
 300 LDY #prompt1 DIV &100
 310 JSR pstring
 320 JSR input
 330 JSR openfile
 340 STA handle1
 350 :
 360 LDX #prompt2 MOD &100
 370 LDY #prompt2 DIV &100
 380 JSR pstring
 390 JSR input
 400 JSR openfile
 410 STA handle2
 420 :
 430 LDY #3:JSR dash
 440 LDY #20:JSR dash
 450 LDA &D0:ORA #2:STA &D0
 460 LDX #window MOD &100
 470 LDY #window DIV &100
 480 JSR pstring
 490 :
 500 LDA #0:STA offset:STA offset+1
 510 LDA #&FF:STA end1:STA end2
 520 :
 530 .main
 540 LDA end1:STA end
 550 LDA handle1:STA handle
 560 LDA #buffer1 MOD 256:STA buffer
 570 LDA #buffer1 DIV 256:STA buffer+1
 580 JSR read80
 590 LDA end:STA end1
 600 LDA end2:STA end
 610 LDA handle2:STA handle
 620 LDA #buffer2 MOD 256:STA buffer
 630 LDA #buffer2 DIV 256:STA buffer+1
 640 JSR read80
 650 LDA end:STA end2
 660 :
 670 JSR display
 680 LDA end1:BPL exit1
 690 LDA end2:BPL exit2
 700 LDA #15:JSR osbyte
 710 JSR osrdch
 720 BCS escape
 730 LDA #12:JSR oswrch
 740 JMP main
 750 :
 760 .exit1
 770 LDX #message1 MOD &100
 780 LDY #message1 DIV &100
 790 JSR pstring
 800 BEQ exit
 810 :
 820 .exit2
 830 LDX #message2 MOD &100
 840 LDY #message2 DIV &100
 850 JSR pstring
 860 :
 870 .exit
 880 JSR out
 890 .close
 900 LDA #0:TAY
 910 JMP osfind
 920 :
 930 .escape
 940 JSR exit
 950 LDA #11:JSR oswrch
 960 LDA #&7E:JSR osbyte
 970 BRK:EQUB 17
 980 EQUS "Escape"
 990 EQUB 0
1000 :
1010 .openfile
1020 LDX #lbuff MOD &100
1030 LDY #lbuff DIV &100
```

```
1040 LDA #&40:JSR osfind
1050 CMP #0:BNE fileok
1060 JMP notfound
1070 .fileok
1080 RTS
1090 :
1100 .notfound
1110 JSR close
1120 BRK:EQUB 0
1130 EQUS "File not found"
1140 EQUB 0
1150 :
1160 .read80
1170 LDY #0
1180 .rloop
1190 STY tempy:LDY handle
1200 JSR osbget
1210 BCS pad
1220 LDY tempy:STA (buffer),Y
1230 INY:CPY #&80:BNE rloop
1240 RTS
1250 :
1260 .pad
1270 LDY tempy:STY end
1280 .ploop
1290 LDA #0:STA (buffer),Y
1300 INY:CPY #&80:BNE ploop
1310 RTS
1320 :
1330 .display
1340 LDA #buffer1 MOD &100:STA buff1
1350 LDA #buffer1 DIV &100:STA buff1+1
1360 LDA #buffer2 MOD &100:STA buff2
1370 LDA #buffer2 DIV &100:STA buff2+1
1380 :
1390 LDX #0
1400 .dloop
1410 LDA offset+1:JSR phex
1420 LDA offset:JSR phex
1430 LDA #32: JSR oswrch:JSR oswrch
1440 :
1450 LDY #0
1460 .dloop1
1470 LDA (buff1),Y:JSR phex
1480 JSR space
1490 INY:CPY #8:BNE dloop1
1500 :
1510 LDY #0
1520 .dloop2
1530 LDA (buff1),Y:JSR checkchar
1540 JSR oswrch
1550 INY:CPY #8:BNE dloop2
1560 :
1570 JSR space
1580 JSR oswrch:JSR oswrch
1590 LDY #0
1600 .dloop3
1610 LDA (buff2),Y:JSR phex
1620 JSR space
1630 INY:CPY #8:BNE dloop3
1640 :
1650 LDY #0
1660 .dloop4
1670 LDA (buff2),Y:JSR checkchar
1680 JSR oswrch
1690 INY:CPY #8:BNE dloop4
1700 :
1710 JSR osnewl
1720 BIT &FF:BPL notesc
1730 JMP escape
1740 :
1750 .notesc
1760 LDA offset
1770 CLC:ADC #8:STA offset
1780 BCC nothi:INC offset+1
1790 .nothi
1800 LDA buff1
1810 CLC:ADC #8:STA buff1
1820 BCC nothi2:INC buff1+1
1830 .nothi2
1840 LDA buff2
1850 CLC:ADC #8:STA buff2
1860 BCC nothi3:INC buff2+1
1870 .nothi3
1880 INX:CPX #16:BEQ dloopend
1890 JMP dloop
1900 .dloopend
1910 RTS
1920 :
1930 .input
1940 LDX #wblk MOD &100
1950 LDY #wblk DIV &100
1960 LDA #0:JSR osword
1970 BCC inputok
1980 JMP escape
1990 .inputok
```

```
2000 RTS
2010 :
2020 .pstring
2030 STX addr:STY addr+1
2040 LDY #0
2050 .psloop
2060 LDA (addr),Y:JSR osasci
2070 BEQ psend
2080 INY:BNE psloop
2090 .psend
2100 RTS
2110 :
2120 .phex
2130 PHA
2140 LSR A:LSR A
2150 LSR A:LSR A
2160 JSR phex2
2170 PLA:AND #&F
2180 .phex2
2190 ORA #ASC"0"
2200 CMP #ASC"9"+1:BCC phex3
2210 ADC #6
2220 .phex3
2230 JMP oswrch
2240 :
2250 .checkchar
2260 CMP #&7F:BCS illegal2
2270 CMP #&20:BCS charok2
2280 .illegal2
2290 LDA #ASC"."
2300 .charok2
2310 RTS
2320 :
2330 .space
2340 LDA #32:JMP oswrch
2350 :
2360 .dash
2370 LDA #31:JSR oswrch
2380 LDA #0:JSR oswrch
2390 TYA:JSR oswrch
2400 LDX #80
2410 .dashloop
2420 LDA #ASC"-":JSR oswrch
2430 DEX:BNE dashloop
2440 RTS
2450 :
2460 .out
2470 LDA &D0:AND #253:STA &D0
2480 LDA #26:JSR oswrch:LDA #31:JSR oswrch
2490 LDA #0:JSR oswrch:LDA #22:JMP oswrch
2500 :
2510 .prompt1
2520 EQUB 31:EQUB 9:EQUB 2:EQUS "File 1 : "
2530 EQUB 0
2540 :
2550 .prompt2
2560 EQUB 31:EQUB 44:EQUB 2:EQUS "File 2 : "
2570 EQUB 0
2580 :
2590 .message1
2600 EQUD &15091F1A:EQUS "End of file 1 "
2610 EQUB 0
2620 :
2630 .message2
2640 EQUD &152C1F1A:EQUS "End of file 2 "
2650 EQUB 0
2660 :
2670 .window
2680 EQUD &4F13031C:EQUW 4
2690 :
2700 .wblk
2710 EQUW lbuff:EQUB 16
2720 EQUB 32:EQUB 255
2730 :
2740 .lbuff
2750 EQUS STRING$(16," ")
2760 :
2770 .buffer1
2780 EQUS STRING$(128,CHR$0)
2790 .buffer2
2800 EQUS STRING$(128,CHR$0)
2810 :
2820 ]NEXT
2830 :
2840 PRINT'"To save object code press COPY ";:
2850 REPEAT UNTIL INKEY(-106)
2860 OSCLI "SAVE DDUMP 900 "+STR$~P%
2870 PRINT
2880 END
```

* Best of BEEBUG *



Applications I Disc

Business Graphics - for producing graphs, charts and diagrams
Video Cataloguer - catalogue and print labels for your video cassettes
Phone Book - an on-screen telephone book which can be easily edited and updated
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
Personalised Address Book - on-screen address and phone book
The Earth from Space - draw a picture of the Earth as seen from any point in space
Page Designer - a page-making package for Epson compatible printers
World by Night and Day - a display of the world showing night and day for any time and date of the year

File Handling for All
on the BBC Micro and Acorn Archimedes
by David Spencer and Mike Williams

Computers are often used for file handling applications yet this is a subject which computer users find difficult when it comes to developing their own programs. *File Handling for All* aims to change that by providing an extensive and comprehensive introduction to the writing of file handling programs with particular reference to Basic.

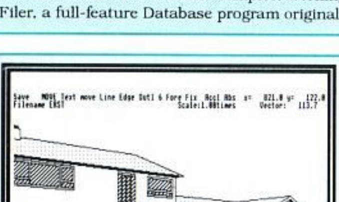
File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File Handling and Databases on the Beeb and the Arc. However, all the file handling concepts discussed are relevant to most computer systems, making this a suitable introduction to file handling for all.

The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical and relational database design, and the book concludes with a chapter of practical advice on how best to develop file handling programs.

The topics covered by the book include:

Card Index Files, Serial Files, File Headers, Disc and Record Buffering, Using Pointers, Indexing Files, Searching Techniques, Hashing Functions, Sorting Methods, Testing and Debugging, Networking Conflicts, File System Calls

The associated disc contains complete working programs based on the routines described in the book and a copy of Filer, a full-feature Database program originally published in BEEBUG magazine.



ASTAAD

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

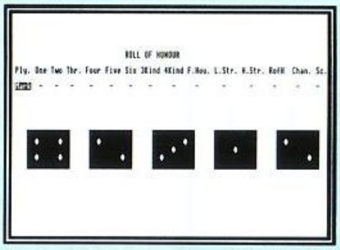
- full mouse and joystick control
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£4.95	**ASTAAD** (3.5" ADFS)	1408a	£4.95
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
Arcade Games (40/80 track DFS)	PAG1a	£4.95	**Arcade Games** (3.5" ADFS)	PAG2a	£4.95
Board Games (40/80 track DFS)	PBG1a	£4.95	**Board Games** (3.5" ADFS)	PBG2a	£4.95

All prices include VAT where appropriate. Please add post & packing (see opposite).

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS.

Board Games


ROLL OF HONOUR
Ply. One Two Thr. Four Five Six 3Kind 4Kind F.Hou. L.Str. H.Str. RofH Chan. Sc.

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.

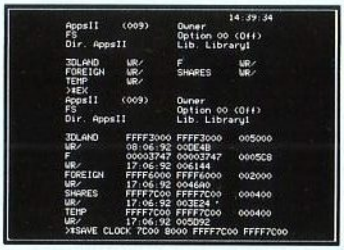
Elevenses - another popular version of Patience - lay down cards on the table in three by three grid and start turning them over until they add up to eleven.

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.



```
                                          14:39:34
AppsII      (009)       Owner
FS                      Option 00 (Off)
Dir. AppsII             Lib. Library1

3DLAND      WR/         F           WR/
FOREIGN     WR/         SHARES      WR/
TEMP        WR/
>*EX
AppsII      (009)       Owner
FS                      Option 00 (Off)
Dir. AppsII             Lib. Library1

3DLAND      FFFF3000 FFFF3000   005000
WR/         08:06:92 000E4B
F           00003747 00003747   0005C8
WR/         17:06:92 006144
FOREIGN     FFFF6000 FFFF6000   002000
WR/         17:06:92 0046A0
SHARES      FFFF7C00 FFFF7C00   000400
WR/         17:06:92 003E24 *
TEMP        FFFF7C00 FFFF7C00   000400
WR/         17:06:92 005D92
>*SAVE CLOCK 7C00 8000 FFFF7C00 FFFF7C00
```

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

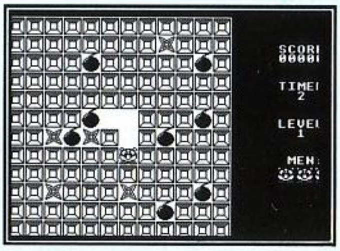
Julia Sets - fascinating extensions of the Mandelbrot set

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.

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?

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.

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

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.

* FINAL * OFFERS *

	Stock Code	Price		Stock Code	Price
File Handling for All					
Book and 40/80T DFS disc	BK04b	£ 9.95	**Filer Database Management**	1417a	£ 4.95
Book and 3.5" ADFS disc	BK06b	£ 9.95	40/80 DFS (*includes home accounts/graphs*)		

POST & PACKING
Please add the cost of p&p when ordering.
When ordering several items use the highest price code, plus half the price of each subsequent code. UK maximum £8.

Post Code	UK, BFPO Ch.I	Europe Eire	Americas, Africa Mid. East	Elsewhere
a	£ 1.00	£ 1.60	£ 2.40	£ 2.60
b	£ 2.00	£ 3.00	£ 5.00	£ 5.50

Tel. (0727) 840303 Fax. (0727) 860263

512 Forum

by Robin Burton

Yet again this month we have a couple of outstanding items on back-up software, plus information on the latest and final updates to David Harper's PD programs, TXTMOUSE and PCCE.

FLEXI-BAK PLUS

I did say a couple of months back that I'd comment on the latest version of Flexi-bak Plus, but until now haven't had the space to do it. However, I've now tried the software in my 512, though not exhaustively, and found problems.

The documentation explains that the latest version of Flexi-bak now uses a form of (I think) LHARC compression. Whether any previous version used compression or not I don't know, but I do know that at least a couple of 512 users have been using an old version of Flexi-bak regularly for some time and like many of the facilities, so obviously this is a fairly new problem.

Whatever the situation with old versions, the changes to (or addition of) compression in the most recent version means that it no longer works in the 512 so far as I can see. As I said, I didn't test it exhaustively, so perhaps an enterprising 512 user has found a fix for the problem, but if so I'm unaware of it and can only warn you of my findings.

PKZIP AGAIN TOO

There are also a couple of final points about PKZIP that it seems weren't stressed enough, so I'll cover those too now.

I have mentioned before that to run PKZIP or UNZIP version 2.04G in the 512 you need PCCE, without which the program crashes. Even so some readers either missed this point or forgot about it, so if you are one of them and have been cursing me for talking about software that doesn't run in your 512 you now know why. The problem is that, unlike earlier versions, PKZIP 2.04G uses an undocumented interrupt function for fast character output to the screen.

Not surprisingly the 512 doesn't provide this routine (it was added in MS-DOS 3, so PCs can have the same trouble), but since PKZIP now uses it by default for all screen output, if you don't avoid the problem a crash is inevitable. It happens as soon as the program tries to display the first character of its name on the screen, so even if you're only expecting the help display it crashes, which is neither encouraging nor enlightening.

Thanks again to David Harper for taking the trouble to find out what's happening in detail and for building an automatic fix into PCCE, more of which later. However, even if you don't have PCCE, you can still run PKZIP 2.04G provided you read the documentation properly.

There is a lot of documentation for PKZIP so I guess most people don't bother reading it. Anyway, back to the problem. You can tell PKZIP not to use the fast character output routine by placing the directive 'PKNOFASTCHAR' in your environment using 'SET', in which case PCCE isn't required. Of course, if you choose this method you'd be best advised to include the statement in your autoexec file so that you don't forget in future, but as usual you don't get something for nothing.

Unlike MS-DOS, in which it expands as required, DOS Plus has a limited and fixed environment space, only 256 bytes in fact. This means that using this space when you can avoid it isn't a good idea, especially if you already have other lengthy environment strings such as a very long path statement and unavoidable specialised directives for other applications.

I must confess that I've never before tried to find out what happens when the 512 runs out of environment space, but I have just done it so that I can tell you. DOS Plus is very well behaved in this case, simply reporting 'Out of environment space', so there's no harm done, although of course the new string you are attempting to add will be ignored.

Environment space isn't likely to be much of a problem for floppy users, but for winchester users it might be. I was surprised to find that in my own system I have a lot less than 100 bytes left for new variables, even though I've always been aware of the limitation and so have avoided lengthy strings wherever possible.

The other point that I must clarify was one that I only brushed on, that of PKZIP back-ups spanning multiple volumes. On checking the documentation for other things I noticed that there's a qualifying statement that this option is only available for DOS version 3 and later, so in fact it won't work in the 512 anyway. Obviously my excuse for not mentioning this fact before is that, not only did I say at the time that I don't use the option, I positively don't recommend it.

PD UPDATES

I had a letter from David Harper a few weeks ago, along with a disc containing updates to his excellent programs PCCE and TXTMOUSE. In fact, this is the second or third update I've had since I offered these programs in FORUM, but these weren't bug-fixes. Like most good programmers David usually can't resist it when he knows there's an improvement he can make to one of his programs, no matter how small the change might be.

In this case though the additions are quite significant. Although the latest changes are not guaranteed to make all DOS software work on the 512, David has addressed certain problems that can be the cause of display or system crashes.

The latest version of PCCE is now 1.22 (so you can check against yours), and two important additions have been made. The first is that multiple display pages have been implemented for modes 0/1 and 2/3. These are the CGA text screen modes which provide 40 or 80 column text in black and white or colour (see Volume.11 No.9, page 36 for DOS screen modes).

Although the 512's implementation of DOS Plus provides these screen modes, in the 512 there's only one display page for each, whereas there are up to seven in a PC depending on mode and colour. The ability to switch display pages allows applications to use multiple displays with a virtually instant swap between them. At the same time alterations to the contents of any one 'screen' (i.e. page) including cursor positioning, whether it's the currently visible display or not, have absolutely no effect on any of the others.

Clearly any program that expects to do this sort of thing in the 512 will be useless, because interrupt calls to switch pages or to write to other pages are simply ignored. They don't cause an error in the application or in DOS, but obviously the 'other' display will never appear. The result is a program that doesn't crash, but which probably doesn't seem to do anything at all.

Most programs of this sort will either fail to change the display at all, or will clear the screen then appear to freeze. In fact

Acorn COMPUTING

will soon be the only magazine that caters for YOU and all BBC Micro machine owners

Are you one of many people looking for a magazine that caters for owners of BBC Micro machines? Let us tell you about Acorn Computing...

Every month we provide comprehensive coverage of education and public domain. Acorn Computing is exactly what you are looking for.

For only £2.95 Acorn Computing is the best buy for owners of any Acorn machines.

Plus...

It gets even better when you subscribe because, in addition to having your reserved copy delivered postage free to your home, subscribers get a special disk containing ingenious and varied new programs written specially for BBC B or Master machines. Every month we have complete programs written specially for you and your computer.

Special subscription offer

To top all of this we have a special subscription offer we have put together for BeeBug readers only. Take out a subscription to Acorn Computing today and we'll send you an exclusive pack of the last six months' subscriber disks for 8-bit machine owners. Not only will you secure interesting reading, useful information and future disks, but you'll also have the last six months' subscriber disks to start off your collection.

This is a great opportunity to save some money, as an ongoing quarterly direct debit subscription costs just £6.73 each quarter, and you get an extra issue each year at no cost.

To SUBSCRIBE and claim your free gift call our 24 hour HOTLINE 051-357 1275 they're waiting for you to select or enter something, but the problem of course is that you can't see the display. Some programs, such as those that use multiple menus, may display the first menu then will fail to respond to a request for a menu (i.e. display page) change.

I've come across quite a few of these programs in my time and they can usually be identified by the fact that if you know which keys to press to exit to DOS (you DO therefore need operating instructions), the operation works perfectly normally, so obviously the program was running properly despite the absence of a visible display.

You can guess from the description that implementing multiple display pages in the 512 wasn't a trivial task. David has had to modify a number of existing ROM BIOS calls and provide several new functions, plus all the appropriate background activities such as memory management.

The second new addition to PCCE caters for programs that read the hardware timer, which they may do for various reasons such as to find out the speed of the processor clock, for example. Essentially such programs repeatedly read a memory location waiting for a change, then note how quickly it happened. Normally such programs never see any change in the 512 so they simply sit and wait for ever. This then is another definite cause of 512 'hang-ups', although in this case it's a real one and re-booting is the only way out. With PCCE many programs that read the hardware timer may now work properly, assuming there are no other problems lurking further in the application.

The change mentioned earlier to allow fast character output was made a few versions ago, but note that this failure may well afflict numerous programs. It's not specifically a PKZIP problem, so again 'across the board' applications compatibility will be improved by PCCE. Since the fix diverts the missing call to an existing routine that does work the overhead is almost nil and it will work for any program. That is why it's a better fix than adding a new application specific environment string, and it's bound to be reliable whenever it's needed.

TXTMOUSE has also now been enhanced, though in this case the changes are less dramatic. While the previous version was restricted to 80 column text screen modes the new program also works in graphics modes. David hasn't added a 40 column text mode but as he says, are there any 40 column mouse driven programs?

If you have an earlier version of PCCE or TXTMOUSE you'd be well advised to update forthwith. In view of its added capabilities, TXTMOUSE is now called the "Master 512 Mouse Driver", but if you just ask for David Harper's mouse driver everyone will know what you mean. PCCE fixes some important shortcomings in the 512's version of DOS Plus and is without doubt a far better program and a better compatibility enhancer than the only alternative. The fact that it's PD and so costs only a couple of pounds while the other used to cost £30 is simply a bonus.

AND FINALLY

And never has this heading been more poignant. I'm sure you're well aware that this is the penultimate 512 Forum. I am, and I've been dreading writing the last one for months. Well, that's my next task. What do you say? How do you say anything suitable after writing for sixty issues?

Watch this space and with a bit of luck I might have some rather surprising information to end on. See you next month.

Cute Key Cuts - in Basic

by David Polak

The program on 'key-cutting' by Chris Robbins in BEEBUG Vol.11 No.8 on page 38 gave an intriguing use of the segments of Wordwise Plus for generating appropriate function key definitions. Now this is inevitably limited by its speed of operation, and not everybody has access to Wordwise Plus. So I thought why not a program to do the same job which works entirely in Basic? Here it is (the program *KeyCut* listed below), fast and convenient, and it goes direct to WW or WW+ on completion, if you wish, with the key definitions ready for use and stored on disc to be invoked using *"Name" or *EXEC."Name" when they are required on another occasion.

A standard file named *KeySafe* is used to hold one set of key definitions, but the program allows different sets of definitions to be made up and stored under user-defined names. These can be used instead of the standard *KeySafe* file.

If this standard file is missing when the program is first run, it will be created and opened when the program is used; otherwise the key definitions stored on disc in *KeySafe* are used each time.

The selected set of definitions is displayed on screen, and any key can be chosen by number (0 - 15) for a new or amended definition.

Definitions are prescribed in the DATA lines. A menu enables a WW function-key operation to be called by its key number, 0-9. Other codes are called by letter, A-R. 'S' is used to allow a string to be included in the definition, with or without apostrophes. This also allows copying from a previous definition displayed at the top of the screen along with a progressive display as the new definition is built up.

A new name can be given under which to store another set of key definitions, and the program ends in either Basic or Wordwise according to choice.

Well there you have it - a simple but effective system which I hope others will find useful.

```
   10 REM Key cutting in Basic
   20 REM Version B 1.0
   30 REM Author  David Polak
   40 REM BEEBUG  March 1994
   50 REM Program subject to copyright
   60 :
  100 ON ERROR REPORT:PRINT" at ";ERL:EN
D
  110 MODE7:PROCinit
  120 REPEAT
  130 PROCnewsafe:PROCshowKeys
  140 REPEAT
  150 K$="":PROCchooseKey
  160 REPEAT
  170 PROCmenu
  180 PRINTTAB(0,24)"More CODE for Key "
;K%;"?";CHR$(131);"Y/N";STRING$(10," ")
  190 UNTIL(GET AND &DF)=78
  200 PROCsetKey
  210 PRINT"Another KEY to code?";CHR$(1
30);"Y/N"
  220 UNTIL(GET AND &DF)=78
  230 PROCspool
  240 PRINT"Another SET of keys?";CHR$(1
33);"Y/N"
  250 UNTIL(GET AND &DF)=78
  260 PROCquit
  270 END
  280 :
 1000 DATA |!  f0 I/O
 1002 DATA |!! f1 Green
 1004 DATA |!" f2 White
 1006 DATA |!# f3 Marker
 1008 DATA |!$ f4 Csr to
 1009 DATA |!% f5 Count to
 1010 DATA |!& f6 Del to
 1011 DATA |!' f7 Del Mkd
 1012 DATA |!( f8 Mve Mkd
 1013 DATA |!) f9 Cpy Mkd
 1014 DATA  "|!, = Csr L"
 1015 DATA |!-  = Csr R
 1016 DATA |!.  = Csr D
 1017 DATA |!/  = Csr U
 1018 DATA |I   = TAB
 1019 DATA |M   = RETN
 1020 DATA |[   = ESC
 1022 DATA |!|\ = CTRL-Csr L
 1024 DATA |!|} = CTRL-Csr R
 1026 DATA |!|^ = CTRL-Csr D
 1028 DATA |!|- = CTRL-Csr U
 1029 DATA |!|L = SHFT-Csr L
 1030 DATA |!|M = SHFT-Csr R
 1031 DATA |!|N = SHFT-Csr D
 1032 DATA |!|O = SHFT-Csr U
 1033 DATA |A   = CTRL-A
 1034 DATA |S   = CTRL-S
 1035 DATA |D   = CTRL-D
 1036 DATA STR.
 1037 DATA END DEFN.
 1030 :
 2000 DEF PROCinit
 2010 A$="0123456789ABCDEFGHIJKLMNOPQRS"
 2020 K=15:S=28:Safe$="KeySafe"
 2030 DIM K$(S),String$(S),Read$(S)
 2040 FOR X=0 TO S:READ K$(X):NEXT
 2050 FOR X=1 TO S
 2060 T$=LEFT$(K$(X),2)
 2070 IF MID$(K$(X),3,1)<>" " T$=LEFT$(K
$(X),3)
 2080 IF MID$(K$(X),4,1)<>" " T$=LEFT$(K
$(X),4)
 2090 String$(X)=T$:String$(0)="|! "
 2100 NEXT
 2110 REM K - No of keys: S - No of stri
ngs
 2120 REM K%(K) - key: K$(K) - string+de
fn
 2130 REM String$(K) - definition only
 2140 CLS
 2150 ENDPROC
 2160 :
 3000 DEF PROCnewsafe
 3010 INPUT'''"Rename key-data file";'"O
R <RETURN> to keep 'KeySafe'",Safe$
 3020 IF Safe$=""Safe$="KeySafe"
 3030 PRINT"Key-data file is now: ";Safe
$
 3040 file=OPENIN Safe$
 3050 IF file=0 ENDPROC
 3060 FOR X=0 TO 15
 3070 Read$(X)=""
 3080 REPEAT
 3090 T$=CHR$(BGET#file)
 3100 IF ASC T$<>&0D Read$(X)=Read$(X)+T
$
 3110 UNTIL ASC T$=&0D
 3120 PRINT".";
 3130 NEXT X:PRINT:CLOSE#file
 3140 FOR X=0 TO 15
 3150 T$=Read$(X)
 3160 T$=LEFT$(T$,(LEN(T$)-1))
 3170 S$=STR$X:OSCLIT$
 3180 NEXT X
 3190 ENDPROC
 3200 :
 4000 DEF PROCshowKeys
 4010 CLS:PRINT'':FOR X=0 TO 15
 4020 PRINT"*Key ";X;"  ";
 4030 OSCLI"*SHOW"+STR$X
 4040 NEXT
 4050 ENDPROC
 4060 :
 5000 DEF PROCchooseKey
 5010 REPEAT
 5020 PRINTTAB(0,21)"(0 - 15)";CHR$(131)
" or <Q> for WW";
 5030 INPUT;T$
 5040 REM PRINTTAB(15,22)STRING$(20,CHR$
(8))
 5050 PRINTTAB(24,21)SPC(8)
 5060 K%=VAL(T$)
 5070 IF T$="Q" PROCquit
 5080 IF K%=0 AND T$<>"0" K%=16
 5090 UNTIL K%>=0 AND K%<16
 5100 ENDPROC
 5110 :
 6000 DEF PROCmenu
 6010 CLS:PRINT''
 6020 PRINTTAB(0,0)CHR$(133)"KEY ";K%;"
now ";CHR$(131);K$
 6030 PRINTCHR$(134)"KEY ";K%;" WAS";CHR
$(130);:OSCLI"*SHOW"+STR$K%
 6040 PRINT:FOR X=0 TO S/2
 6050 IF X<10 PRINT;X;TAB(3);K$(X);
 6060 IF X>9 PRINT;CHR$(X+55);TAB(3);K$(
X);
 6070 IF X<S/2 PRINTTAB(19);
 6080 IF X<S/2 PRINTCHR$(X+56+S/2);" ";K
$(1+X+S/2)
 6090 NEXT:PRINT
```

Continued on page 38

BEEBUG Education: Earthwarp

Reviewed by Mark Sealey

Product	**Earthwarp**
Publisher	**Longman Logotron** **124 Cambridge Science Park,** **Milton Road, Cambridge CB4 4ZS.** **Tel: 0223 425558** **Fax: 0223 425349**
or	**BBC Educational Publishing** **Freepost LS 2811,** **PO Box 234, Wetherby,** **West Yorkshire LS23 6YY.**
Prices	**£28.20** **£5.99 both audio cassette and pack of 5 storybooks** **£6.25 photocopy masters** **£2.99 novel**

All prices inclusive of VAT where applicable.

Just when you thought it was safe to think that there'd be no more releases of 8-bit software (this is the last Beebug Education when all is said and done), along comes *Earthwarp* from Longman Logotron.

TV TIE-IN

Like the subject of the very first Beebug Education seven years ago (the Domesday Project), Earthwarp is closely associated with the BBC; this is fitting since the Corporation, of course, played such a major part in the history of our computer system.

This time the tie-in is with the *Look and Read* TV series, a new cross-curricular programme being broadcast this term. *Look and Read* also spawned *Through the Dragon's Eye*, *Geordie Racer* and *Sky Hunter*, all of which have led to BBC software. Indeed, this software is only one item in a group of products including photocopiable sheets, a book and cassette etc. It is also available for the Acorn 32-bit series for the same price.

THE SCENARIO

Since the software closely reflects the story on TV, the scene must be set. The main character of Earthwarp is Ollie, a traveller from Gia, a distant planet in another solar system. His planet was once like ours, though, but its inhabitants took insufficient care of it such that it would now be a long way down your list of places to visit once space travel becomes an option for the package holiday.

Ollie and his sister Ellie are devoted to travelling around the universe to help other people take care of their planets and so prevent them from meeting a similar fate.

They have to visit earth to find a probe installed previously to monitor pollution; if they don't, it will surely explode destroying the seaside town of Southbeach, the location for this software.

This is the task, the main thrust of the adventure. Ollie has three friends from earth, the children Amina, Jenny and Martin; they know enough about earth and its ways to help Ollie, who is a stranger here.

SIMULATION

The publishers call Earthwarp a 'simulation'; not really, but a persuasive and convincing attempt to involve the pupil-users (probably from key stages 2 and 3) in a series of problem-solving situations which sponsor purposeful and challenging activities in several curriculum areas: language, maths, science, technology, geography, music and mathematics. Complete these successfully and the planet is saved!

In essence, Earthwarp presents the pupils (who could work in a variety of ways - in groups, as individuals, or even as a class) with a set of tasks connected with the story just outlined. The main message, of course, is an environmental one.

The tasks include wordsearches, work with co-ordinates, spatial puzzles, sequences, word puzzles, factual stimuli on the solar system, all linked thematically yet each valid in its own right. Well, if simply, designed too.

EASY TO USE

These are all reached by a menu containing some dozen or so tasks: Longman Logotron has packed a lot onto this double-sided 40 track DFS disc. So it is possible to enter the program at any point and not to have to repeat material already covered at second and successive visits to the computer.

Control of the software is simple and intuitive. Use of keys and your passage through the software is natural and presents no difficulties.

There is a very straightforward (laboured, even) 'tutorial' at the start, though this can be skipped; an introduction to the keyboard and the conventions used in the program, moving on, confirming answers and so on. In this respect, Earthwarp is as easy to use as any software at this stage in the game.

DOCUMENTATION

With the disc comes a thirty-page A4 book. This provides everything you'd need to know - even as a beginner new to computing - about how to find ways into the software from a curriculum point of view, how to navigate around it once up and running, and how to set up activities away from the computer, particularly in environmental work.

Links to the National Curriculum are set out and the relevant Attainment Targets tabulated with references to the relevant section of the software. These are presented twice: firstly as ATs listed by Earthwarp activity, and then with a fuller statement of this week's version of the National Curriculum.

Very many suggestions for other activities are given, as are sources for environmental organisations (though, oddly, neither Friends of the Earth nor Greenpeace), books, programmes and programs as well as the odd article and several items of computer and non-computer equipment.

CONCLUSION

All in all, Earthwarp will meet all the needs that it is designed to do. It will provide enough challenging and well-thought out material in an integrated way, across the curriculum but with a unifying theme that those who use it will see and appreciate. The tried and tested idea of tying software in with a TV series is a good one. This software from experts Longman Logotron does it justice and can be thoroughly recommended.

POSTSCRIPT

It's getting late; probably most of what has to be said about the BBC computer and its software for children has been said by now. Some of those pupils who were working, say, at what we now call key stage 3 when this column began in 1987 are old enough to be successful programmers by now.

Only a small minority could have predicted the changes in education experienced since then. That said, probably an even smaller minority could begin to imagine what we shall be excited about in seven years' time.

Three things are certain: advances in techniques of mass digital storage are likely to blur still further the distinction between computers, phones, faxes, video and indeed most forms of communication, including TV.

The issues of cross-platform movement of data are likely to have largely faded and ceased to matter.

And - just as has always been the case - the equipment will be manipulated, manoeuvred and distorted to reflect the latest trends in educational and social thinking.

You should be clear, though, that the changes and advances that will be made won't be made on a BBC B or Master. If you've put it off until now, think long and hard about upgrading; raise the money any which way. And make no mistake, either, that as these changes continue to occur, they will alter radically the way schools function and are organised.

For now there are relatively few sources for those of you who wish to hold on to your BBC for a while yet. Beebug magazine remains the first among equals; much of the material remains relevant, stimulating and worth retaining, and it speaks volumes for the quality of the platform that the forum has lasted as long as it has.

The dedicated educational magazine Educational Computing and Technology (contactable on 0895 622112) has also begun (yes that's right, begun) a column specifically for 8-bit users. Worth looking at.

On a personal note, I have enjoyed immensely writing this column for Beebug all these years: some three dozen issues. Before signing off, I extend many thanks to Mike Williams and the staff in St Albans for the support and help they have given throughout. What a community to be a part of, and if what you have read has proved in any way useful, then that's good.

Cute Key Cuts - in Basic (continued from page 35)

```
 6100 PRINT'CHR$134"For key ";K%;" SELEC
T";
 6110 PRINTCHR$(131)" ( 0 - 9 or A - S )
"
 6120 B=0:REPEAT
 6130 B=B+1:IF B>1 VDU7:PRINTTAB(0,22)CH
R$(131)" ( 0 - 9 or A - S ) please"
 6140 C=0:REPEAT
 6150 C=C+1:IF C>1 VDU7:PRINTTAB(0,22)"S
elect again 0-9 or A-S"
 6160 A%=INSTR(A$,GET$)
 6170 UNTILA%
 6180 A%=A%-1
 6190 T%=55:IF A%<10 T%=48
 6200 PRINTTAB(0,22)CHR$(134)"..";CHR$(A
%+T%);CHR$(130);" ";String$(A%);CHR$(134
)" OK? Y/N";STRING$(10," ")
 6210 UNTIL(ASC GET$ AND &DF)=ASC"Y"
 6220 IF A%<28 K$=K$+String$(A%) ELSE IN
PUT"String: ";T$
 6230 IF A%=28K$=K$+T$
 6240 PRINTTAB(0,0)CHR$(133)"KEY ";K%;"
now ";CHR$(131);K$
 6250 ENDPROC
 6260 :
 7000 DEF PROCsetKey
 7010 OSCLI"*KEY"+STR$K%+K$
 7020 CLS:PROCshowKeys:PRINT
 7030 ENDPROC
 7040 :
 8000 DEF PROCspool
 8010 CLS:PRINT''"SPOOLING ";Safe$;" - p
lease wait"
 8020 OSCLI"*SP."+Safe$
 8030 PRINT:FOR X=0 TO 15
 8040 PRINT"*Key";X;
 8050 OSCLI"*SHOW"+STR$X
 8060 NEXT
 8070 *SP.
 8080 ENDPROC
 8090 :
 9000 DEF PROCquit
 9010 PRINT"<RETURN> to go to Wordwise"
 9020 A%=GET:IF A%=13 THEN OSCLI"*WORD."
 9030 *FX15,0
 9040 ENDPROC
```

Machine Code Corner

In which Toad gets his ROMs spliced.

Well, Toad fans, here we are at the penultimate issue - it's crept up on me unawares (have you ever had your unawares crept up on?) - and suddenly there are all these topics I always meant to tackle but put off until another day, and only two issues left to do them in. For this reason this month's offering is slightly 'bitty'; hope you don't mind, but Mr T has a few things he wants to get off his ventral surfaces (or, in your terms, chest).

Here's one such 'bit' which has always caught my imagination. Have you ever noticed that certain assembler mnemonics are also words? BRA, for example, STY (oink) and TAX, and the river TAY, if that's not cheating. I discount AND and BIT, which are just English words with their usual meanings, more or less. Here's this month's competition: find one German word, one French and three Latin words in the instruction set - four in Latin if you count the Master-only mnemonics. To start you off, I'll give you that NOP is Dutch, Flemish and Afrikaans for the 'nap' or 'pile' of a carpet. There must be others in other languages which Mr T doesn't know - what about Spanish, or the Slavonic lot? DEC, EOR and INC sound encouraging... but I'd hate to have to learn a language which contained words like TSX or PLP.

One major subject, which I always meant to deal with in a separate article, is joining sideways ROMs together. Over the years, BEEBUG has featured so much useful software in ROM format that there must be many of you out there who put in your floppy every time and watch two or three sideways ROMs load from separate files into separate slots. Even the biggest ROMs listed in BEEBUG are unlikely to run to much over 3K of assembled code - none of mine ever went over two - and your four precious slots can hold 16K apiece. So a fun project for a rainy afternoon is to combine the ones you use most into a single block of code occupying just one file on disc and one SRAM slot.

I'm not talking here about language ROMS, of course, but service ROMs whose type-byte is &82. Nor do I mean commercial ROMs like the Beebug Basic Booster or Disc Doctor. These major commercial products tend to fill up quite a lot of their space anyway, but if you did want to combine two shortish ones you'd need the assembly texts of both. You would be very hard pushed to alter and relocate any large amount of object code: human beings aren't cut out to do that, which is why the Lord gave us assemblers. With the assembly texts, however - which is what our dear old mag provides - the job is easy enough.

You need to make one important decision at the start. You could join the assembly texts together and run them as one program to produce your combined block of code. On the other flipper, will you alter the first block, run it, note down some numbers, load in the second, alter it and run it to put its object code into the slot after the first lot? The first way can create problems: you might run short of memory - I've always thought it really silly that we have to assemble our object code to main memory at O% before moving it into the SRAM slot, memory which could have been used for assembly text. I know that ways around this have been found, but I've never seen one and my project to develop one never really got under way.

But I digress. (What? Really? - Ed.) The second snag which can crop up with combined assembly texts is that you might get a duplicated label; if a particular label is used in both texts then on the second pass of the assembler through the amalgamated text, *all* references to it will be to the last-mentioned version. Then again, you've got do lots of little chores like getting rid of the FOR ... TO ... [OPT (i.e. the start of the OPT loop and the opening bracket of the second listing) and harmonising the location of O% and the arrangements for the SRWRITE at the end. Sorry to be infuriating, but I can't give you *exact* instructions for this, it's just a question of a common-sense look at the consequences of amalgamating two chunks of Basic.

Despite all this, joining the texts can be fun and has a more human 'feel', to me, than the cold and impersonal 'feel' of running separate programs. No, I haven't flipped my griblets (Are you sure? - Ed.), for programs and computers, like literature and music, have distinct styles and 'feels', as sure as my name is Montague Aethelfrith de Toad.

To join two assembly texts, or indeed any two Basic programs, load the first one and do PRINT~TOP. Subtract two from the value of TOP. Yes, you can take two away from even an 'awkward' Hex number if you think about it. Don't give in and do PRINT~TOP-2 or you'll never get any practice. You know Mr T's attitude - don't let Hex beat you, master it and then get the satisfaction of using it with fluency. Anyway, then just do *LOAD <secondprog> 5678, or whatever number TOP-2 turned out to be. Renumber it and tackle all the little 'houskeeping' tasks I mentioned just now, then you're ready to modify the actual assembler. We'll come to that bit in a minute.

To do it by running the two listings one after the other, load in the first and modify it as described below, run it and note down the *final* value of *P%*. Then load the second text, modify it and also alter the *starting* value of *P%* from &8000 - which is the only value it could have had - to the value you just noted down. You'll probably also have to change this figure - again, from a starting value of &8000 - in the *SRWRITE, which is probably at the end. Or the programmer may have set another variable to &8000 at the start and then set *P%* and the *SRWRITE from that.

Now to the modification of the actual ROMs, regardless of which method of combining them you choose. The basic idea is very simple: locate the RTS in the first one which returns to the current language with the call *UN*claimed (see my articles in BEEBUG Vol.11 No.6 onwards), and then replace it with a JMP to the start of the next ROM. Now, programmers generally push all the registers at or near the start of their ROMs, so both exits are invariably marked by the pulling of all three, thus: PLY:PLX:PLA (not necessarily in that order):RTS - or, if for the steam-powered Beeb, something like PLA:TAY:PLA: TAX:PLA:RTS. Those are the 'no-claim' exits. Now 'claiming' the call is what you do when your ROM has recognised a star-command and done its stuff. 'Claiming' just means setting the accumulator to zero to signal to the MOS that it can stop offering the call to all the ROMs, because you've dealt with it. In this situation you don't want your second ROM to do anything, so any exit which has LDA #0:RTS, almost certainly after pulling the registers as above, should be left alone. Now add a label at the very end of the code, last thing before the closing bracket. You'll then remove that bracket and its associated NEXT if using method one. The label *.end* will do fine (one of Mr T's wittier and more culcherd offerings, doncha think?) but do LIST IFend to check that *.end* hasn't been used before. Find the no-claim exit - or possibly exits, you occasionally find more than one - and replace the RTS by 'JMP end'.

The more obvious effect of this is that star commands rejected by the first ROM will be examined by the call-checking routine of the second, which will deal with them exactly as it would have done had it been on its own. The less obvious effect is that both ROMs will get all the 'unseen' calls which the MOS issues to alert all ROMs to events going on in the outside environment, to offer them the opportunity to reserve various types of workspace in main memory and so on. These calls are never claimed, of course, because if you claim them none of the ROMs below you in the pecking-order will receive them.

Just two of these never-claimed calls have effects visible to the user, and those are the *HELP commands. If you identify and deal with the exits correctly, *HELP will produce replies from each ROM separately, as if they were on their own. To me this is fine, but if you don't like it you can easily find and modify the help messages within each assembly text. The other type of *HELP call is the one where you use the prompt supplied by some ROMs in response to the ordinary *HELP. Calls like *HELP DFS or *HELP MOS. These should also work if you've dealt correctly with the exits.

What do you do to the second ROM? Only one change is essential, and that's to get rid of the three zeros which always constitute the start of a service ROM. I always code them as BRK:BRK:BRK; others sometimes write EQUW 0:EQUB 0 or some such. Anyway, get rid of the first three bytes. In fact, you could get away without even that if you wrote JMP end+3 in the first text. The next statement is 'JMP somewhere'; the label *.somewhere* will be further inside - maybe a lot further inside - and will mark the routine which sorts out the calls. Leave that JMP in for now. After this come the type-byte, the copyright-string offset, then the title and binary version-string for the *ROMS call, none of which are of any use now, so if you're reasonably confident, hack them out.

Incidentally, we've seen that *HELP calls will be OK, but *ROMS will only print the title and version-number of the first ROM it finds: it prints everything from &8008 to the next zero as a string, then a space, then the byte at &8007 as a Hex number. There's nothing you can do to change that, so you'll want to alter the title of the first ROM to take account of the new addition - and the *ROMS routine won't accept carriage returns. Sorry about that; I'll leave you to sort it out. Then the copyright string - that can go, too. ('Ahem... my client was acting on the instructions of a toad, M'lud.' 'The advice of reptiles, and indeed of gerbils, is no defence in law, Mr Whoople. This is a very serious breach of copyright. Fined seven shillings and fourpence.') After the copyright string should be the first bit of useful code; if it's prefixed by the label referred to by the initial JMP, then you can scrap both JMP and label - you needn't jump to where you already are - but often the first routine written is not the start routine, in which case leave the JMP as the only part of the header not deleted.

And that should be it. Having run either a combined text or the two separately, (carefully saved to disc!) note the final value of *P%* and test thoroughly. If all is well, *SRSAVE the new ROM with a new filename, bask in the glow of a job well done and bore the wife, kids, workmates, goldfish, etc. for weeks with tales of your legendary skill as a programmer and your close brushes with the dreaded machine-crash.

So now... sob... we've come to the last-ever 'next month' preview. Here... sob... it is: sniff... sob... Next month, Mr T reveals the mystic secrets of the ancient sages: Beebs in the hieroglyphics of the pyramids, and how mummified Acorn Electrons buried with their owners will still run their mysterious programs today.

1st course

BBC Basic - A New Beginning

Marshal Anderson looks at how we got here, and, occasionally, why.

In this penultimate article for the First Course column I thought it would be a nice idea to stroll back through the history of the language you are currently using.

The Beginners All Purpose Symbolic Instruction Code was first developed in America at Dartmouth Collage, New Hampshire in 1964 as a simple computer language for introducing computer programming and also as a tool to give professionals in certain areas very flexible access to the mathematical power of the computer. The idea was that the user did not need to get their hands dirty with machine code, or anything like it, and that they could not do anything drastically wrong and crash the machine.

Basic was designed as an *interpreted* language, that means that the computer reads the instructions one line at a time, converts them into machine code, acts on them then forgets them. The advantage of this, especially to the beginner, is that Basic can halt the program at the line where an error has occurred and tell you what's wrong. The disadvantage is that this constant interpreting and re-interpreting of the same lines slows things down a lot. The opposite to interpreted languages are compiled languages. This is where the whole program is converted to machine code before it is run. This makes it fast, but it's very difficult to track down errors.

It's important to keep in mind that the language was not created for writing serious software. However, over time, its ease of use was combined with the growing power of the hardware until it became quite a practical language for many different tasks.

The first thing to remember about BBC Basic is that, for its time, it was a huge advance on the other versions of the language around. Okay, it didn't have some of the gimmicky stuff that Sinclair offered, like single key command entry, or even the more serious stuff, like error checking on entry, but it was a serious step forward. The problem for us programmers was that we didn't quite understand what it was we had our hands on.

Now, this is mostly due to the nature of programming, especially at the hobbyist level, and at that time hobbyist was all there was. We look for what we know, what is familiar to us, find it, use it and stop. You may not believe me but look at the development of software over the years and you'll see that it took programmers a long time to get to the potential of the machine - I'm not at all sure we've reached it yet.

To see just what a leap forward BBC Basic was from the original I went back to a key book on Basic written in the 70s; 'The Basic Cookbook' - now I know why I kept all this junk! Although Basic was, at that time, over 10 years old, most machines still stuck to the conventions of the original so it was possible to produce such a generic tome.

All the books at the time were based around the idea that programs would be run on mainframe computers, probably remote from the user - these were slow machines indeed by today's standards.

Input and output was by a teleprinter arrangement and you could not expect the computer to run your program immediately, you might well have to wait until the following morning for a result - a long time just to find a syntax error.

As the language was implemented on smaller machines some commands disappeared from it. The most noticeable example were the matrix commands, prefixed MAT. These applied to whole arrays and were obviously meant for number crunching. It's interesting to note that they are re-appearing now that machines are powerful enough to deal with them. There was also a set of commands to deal with signing on and off the mainframe. But, much more important than this, and the thing that suddenly made the Basic Cookbook and its companions out of date, was the advent of graphics. Before 1981 or so, computer graphics was the process of producing pictures of Snoopy by over-printing letters.

This is where BBC Basic really did the business. Its main rival at the time was the Spectrum, along with such forgotten beasts as the Vic 20 and the Dragon. The language designers in all these machines were rushing to keep up with hardware developments. Suddenly they found they had to deal not only with proper graphics, they also had to deal with colour and sound. This led to a lot of half measures and short cuts; many of these required a programmer to access memory directly which defeats the object of Basic and could cause huge problems with machine crashes. In the end, and it is a fairly subjective view, Acorn's designers got it right. The MOVE and DRAW commands were simple but the clever bit was PLOT, not only flexible but extendible. This, along with the VDU commands gave the programmer a great degree of control over the screen but left Basic in charge of the memory - a bit slower but infinitely safer and in the spirit of Basic. It's a measure of the success of this initial approach that Basic V on the Arc still uses these commands.

Sound was a more difficult thing to deal with and no version of Basic at the time, or indeed now, managed to make this simple. BBC Basic, however, offered four channels and the ENVELOPE command which meant that there was a huge range of sounds available and that even speech was possible without extra hardware.

The sound and graphics commands, good though they were, were responses to developments in hardware, they were not changes in the nature of the language itself. Underneath all the flashing lights and clever noises something very much more subtle, but just as important, was going on.

IT'S STRUCTURED

Basic, as a language, is linear - it must be because it's got line numbers. Those of us wedded to Basic perhaps have some trouble with the idea of a language with no line numbers; how does the program know where to start or what comes next?

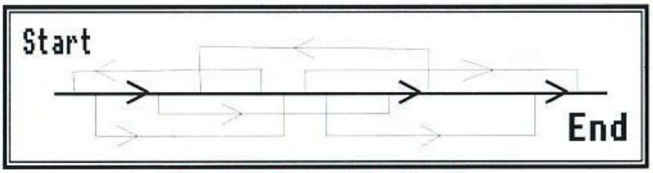


Figure 1. Basic looks like this

Basic, in its original form, encourages an approach to programming which was probably fine for the short pieces people were writing in 1964, but which causes a lot of problems when you come to a more substantial program. What you have is a long line of statements and a set of commands that let you jump about on that line - especially the much frowned upon GOTO. A computer program usually spends a lot of time making decisions and then taking different actions based on those. Trying to achieve this with GOTOs means that the flow of the program is constantly interrupted, the action is jumping about all over the place. Consequently, in any program of any length, it soon becomes impossible to keep track of the processes.

However, many powerful languages, like C for instance, don't need line numbers because they are structured. Programs are developed in free(ish) standing blobs which call each other, the user simply calls the first blob. The important point is that the blobs have names, rather than numbers, and this does something to your head. It makes you think in a structured way where the relationship between blobs is vital but their order unimportant.

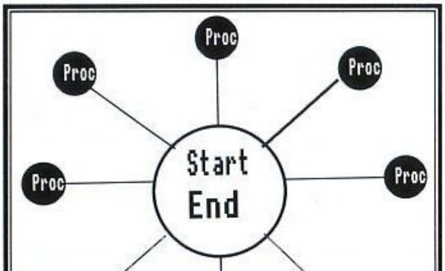


Figure 2. BBC Basic - A blobby approach

Basic, in its earlier form, was able to go some way towards the blobby approach. The commands GOSUB and RETURN allowed a programmer to set up subroutines away from the main code to handle frequently used operations. The interesting thing about this is that, in the tutorial books of the time, only very trivial examples are given. You might see subroutines to print frequently used text or scroll the screen, but there was no attempt to write a program entirely using GOSUBs, indeed to write the subroutines *first* and then string them together in a whole program.

BBC Basic opened the door to properly structured programming, though it took us a while to notice it, with functions and procedures. Instead of writing a program starting at line 10 with PRINT "Hi there", you could start to think of the structure of your program and write it in procedures that had meaningful names. Names are important, as you developed a program the normal re-numbering that would take place would constantly 're-name' your GOSUBs by giving them different line numbers. More than that, subroutines aren't self-contained and sealed off from the rest of the program in the way procedures are. The use of local variables in procedures means that you can write your procedure absolutely *knowing* that it will not accidentally make a dogs breakfast of some other cherished routine that was working so well you forgot all about it. What's more, you can add *arguments* to procedures so that what you are really doing is adding new commands to Basic.

Procedures changed the way we looked at Basic, they even changed the way Basic looks to us. Suddenly things were tidy and organised on the printout, bugs easy(ish) to nail and our prize good bits could be easily transferred to other programs. The stepwise approach it encouraged makes large projects much less daunting and the speed of the machine makes the resulting programs thoroughly usable.

But there's more, and we'll look at some of it in next month's final column.

Graffer (Part 2)

The plot thickens as George Crossly concludes his graph plotting package.

Last month I gave you the full *Graffer* program, but as it stands, it is a little cumbersome to use. This month I am completing the package with a series of subsidiary programs to help make things easier.

All the following programs should be saved in the same directory as Graffer itself. *Newfile* and *Adddata* expect to find subdirectories F and T.

Newfile and Adddata let you to generate and modify data files for use with Graffer. When you've responded to the prompts and confirmed that you're satisfied with the result, your new or modified data file will be saved in directory T or F, depending on whether you've declared it to be a timefile or not.

Dump2 dumps the mode 128 screen to an Epson LX86 printer in low speed double density mode. Graffer calls Dump2 from line 2260 so it must be saved as *Dump2*. As it uses an assembler routine to speed up the process, printing speed is limited by the printer. The program needs no user input.

Savescreen is a modified version of a program which appeared in BEEBUG Vol.5 No.9. It saves a good deal of swapping from dump routine to plotting program if you're printing a number of graphs. As it uses mode 0 it doesn't destroy your graph on the mode 128 screen. The program can be used to load the files it produces back to either screen - load to the shadow screen if you're going to dump the result to your printer using Dump2. Savescreen is called by Graffer at line 2250 and so must be saved as *Savescreen*.

This completes the package and gives you some powerful tools for creating graphs, I hope you find them useful.

Listing 1

```
   10 REM Program Adddata
   20 REM Version B1.2
   30 REM Author  George Crossley
   40 REM BEEBUG  March 1994
   50 REM Program subject to copyright
   60 :
  100 MODE 131
  110 CLS
  120 *CAT F
  130 *CAT T
  140 PRINT:INPUT "Input Directory.Filen
ame: " A$
  150 X=OPENUP A$
  160 PRINT '"Do you wish to read the fi
le?"' : I=GET
  170 IF I=89 OR I=121 THEN PROCreadfile
  180 INPUT#X,N,TIM
  190 PRINT '"Do you wish to change any
of the existing data? " : I=GET
  200 IF I=89 OR I=121 THEN REPEAT : PRO
Cchange : UNTIL I=89 OR I=121
  210 INPUT '"How many more points do yo
u wish to add? " M
  220 IF M=0 THEN CLOSE#X : END
  230 PTR#X=0
  240 PRINT#X,N+M
  250 REPEAT
  260 PTR#X=PTR#X+1
  270 L=EOF#X
  280 UNTIL L=-1
  290 IF TIM=0 THEN PROCffile ELSE PROCt
file
  300 CLOSE#X
  310 END
  320 :
 1000 DEF PROCffile
 1010 DIM T(M,1)
 1020 REPEAT
 1030 FOR A=1 TO M
 1040 PRINT "INPUT X";A+N", Y";A+N; : IN
PUT "  " T(A,0), T(A,1) : NEXT A
 1050 PRINT"Accept these? (Y or N)":P=GE
T
 1060 UNTIL P=89 OR P=121
 1070 FOR B=1 TO M
 1080 PRINT#X,T(B,0) : PRINT#X,T(B,1) :
NEXT B
 1090 ENDPROC
 1100 :
```

```
1110 DEF PROCtfile
1120 DIM T(M,1)
1130 REPEAT
1140 FOR A=1 TO M
1150 PRINT "INPUTY";A+N; : INPUT "  " T
(A,1) : T(A,0)=A+N : NEXT A
1160 PRINT"Accept these? (Y or N)":P=GE
T
1170 UNTIL P=89 OR P=121
1180 FOR B=1 TO M
1190 PRINT#X,T(B,0) : PRINT#X,T(B,1) :
NEXT B
1200 ENDPROC
1210 :
1220 DEF PROCreadfile
1230 PTR#X=0
1240 INPUT#X,N,TIM : PRINT '"N=";N,"TIM
=";TIM'
1250 A=0
1260 REPEAT
1270 INPUT#X,J,K : PRINT "X";A;"=";J,"Y
";A;"="K
1280 A=A+1
1290 L=EOF#X
1300 UNTIL L=-1
1310 PTR#X=0
1320 ENDPROC
1330 :
1340 DEF PROCchange
1350 INPUT '"Which record number? " H
1360 PTR#X=12
1370 PTR#X=PTR#X+(12*H)
1380 IF TIM=0 THEN PRINT "INPUT X";H;",
Y";H; : INPUT "  " F,G : PRINT#X,F,G
1390 IF TIM=1 THEN PRINT "INPUTY";H; :
INPUT "  " G : PRINT#X,H,G
1400 PROCreadfile
1410 PRINT '"Accept these now?" : I=GET
1420 ENDPROC
```

Listing 2

```
 10 REM Program Dump2
 20 REM Version B1.1
 30 REM Author EVALM.J.Crossley
 40 REM BEEBUG  March 1994
 50 REM Program subject to copyright
 60 :
100 MODE 0
110 DIM DUMP% 150
120 P%=DUMP%
130 VDU 21
140 [
150 OPT 0
160 STX &78
170 STY &79
180 LDA#&6C:LDX#&01:LDY#&00:JSR&FFF4
190 LDY#7
200 .TRANSFER LDA(&78),Y:STA &70,Y
210 DEY:BPL TRANSFER
220 LDY#7
230 .MAKECHR LDX#7
240 LDA#1:JSR &FFEE
250 .SHIFTOUT ASL &70,X
260 ROR A
270 DEX:BPL SHIFTOUT
280 JSR &FFEE
290 DEY:BPL MAKECHR
300 RTS
310 ]
320 VDU 6
330 Z%=HIMEM
340 VDU 2
350 VDU 1,27,1,51,1,24
360 FORV%=1 TO 32
370 VDU 1,27,1,42,1,1,1,640 MOD &100,1
,640 DIV &100
380 FORH%=1 TO 80
390 X%=Z% MOD &100:Y%=Z% DIV &100
400 CALL DUMP%
410 Z%=Z%+8
420 NEXT H%
430 VDU 1,13
440 NEXT V%
450 VDU 1,27,1,64
460 VDU 3
470 *FX 3 0
480 END
```

Listing 3

```
 10 REM Program Newfile
 20 REM Version B.1
 30 REM Author  George Crossley
 40 REM BEEBUG  March 1994
 50 REM Program subject to copyright
 60 :
100 MODE 131
110 PRINT "Is this to be a TIMEFILE? "
;:X=GET
120 IF X=89 OR X=121 THEN TIM=1 ELSE T
IM=0
130 PRINT:INPUT "Input file name: " A$
140 IF TIM=1 THEN *DIR T
150 IF TIM=0 THEN *DIR F
160 X=OPENOUT A$
170 IF TIM=1 THEN PROCtfile ELSE PROCf
file
```

```
 180 CLOSE# X
 190 *DIR ^
 200 END
 210 :
1000 DEF PROCffile
1010 REPEAT
1020 INPUT "No. of data points (50 max.
) "N
1030 UNTIL N>1 AND N<51
1040 PRINT#X,N,TIM
1050 DIM T(N,1)
1060 T(0,0)=0 : T(0,1)=0
1070 REPEAT
1080 FOR A=1 TO N
1090 PRINT "INPUT X";A", Y";A; : INPUT
"  " T(A,0), T(A,1) : NEXT A
1100 PRINT"Accept these? (Y or N)":P=GE
T
1110 UNTIL P=89 OR P=121
1120 FOR B=0 TO N
1130 PRINT#X,T(B,0) : PRINT#X,T(B,1)
1140 NEXT B
1150 ENDPROC
1160 :
1170 DEF PROCtfile
1180 REPEAT
1190 INPUT "No. of months/years (max. 1
2) " N
1200 UNTIL N>1 AND N<13
1210 PRINT#X,N,TIM
1220 DIM T(N,1)
1230 T(0,0)=0 : INPUT "Input initial Y
value " T(0,1)
1240 REPEAT
1250 FOR A=1 TO N
1260 PRINT "INPUTY";A;:INPUT "   " T(A,1
):T(A,0)=A:NEXT A
1270 PRINT"Accept these? (Y or N)":P=GE
T
1280 UNTIL P=89 OR P=121
1290 FOR B=0 TO N
1300 PRINT#X,T(B,0) : PRINT#X,T(B,1)
1310 NEXT B
1320 ENDPROC
```

Listing 4

```
  10 REM Program Savescreen
  20 REM Version B2.00
  30 REM Adapted by George Crossley
  40 REM BEEBUG  March 1994
  50 REM Program subject to copyright
  60 :
 100 MODE 0:REM Shadow screen is preser
ved on change to non-shadow mode
 110 VDU5
 120 PROCchoice
 130 VDU4
 140 END
 150 :
1000 DEF PROCchoice
1010 GCOL3,1:PROC1s:A$=GET$:PROC1s
1020 IF A$="S" THEN PROCsave
1030 IF A$="L" THEN PROCload
1040 ENDPROC
1050 :
1060 DEF PROCsave
1070 PROCfilename
1080 CL$="SAVE "+fn$+" 3000 8000"
1090 GCOL3,1:PROC1t:B$=GET$:PROC1t
1100 IF B$="S" THEN *FX108,1
1110 PROCoscli(CL$)
1120 IF B$="S" THEN *FX108,0
1130 ENDPROC
1140 :
1150 DEF PROCload
1160 PROCfilename
1170 CL$="LOAD "+fn$
1180 GCOL3,1:PROC1u:B$=GET$:PROC1u
1190 IF B$="S" THEN *FX108,1
1200 PROCoscli(CL$)
1210 IF B$="S" THEN *FX108,0
1220 ENDPROC
1230 :
1240 DEF PROC1s
1250 MOVE 50,1000:PRINT "L - LOAD    S
- SAVE"
1260 ENDPROC
1270 :
1280 DEF PROCfilename
1290 GCOL3,1
1300 MOVE 50,1000:INPUT"NAME:"fn$
1310 MOVE 50,1000:PRINT"NAME:";fn$
1320 ENDPROC
1330 :
1340 DEF PROCoscli(CL$)
1350 $&900=CL$:X%=0:Y%=9:CALL&FFF7
1360 ENDPROC
1370 :
1380 DEF PROC1t
1390 MOVE 50,1000:PRINT "S - SAVE SHADO
W SCREEN   M - SAVE THIS SCREEN"
1400 ENDPROC
1410 :
1420 DEF PROC1u
1430 MOVE 50,1000:PRINT "S - PRINT TO S
HADOW SCREEN  M - PRINT TO THIS SCREEN"
1440 ENDPROC
```

Fish (continued from page 8)

```
 2380 p%=fish%(X%):ch(pos%)=p%
 2390 IF p%<5 eye%=-66 ELSE eye%=96
 2400 MOVE fx%,fy%:PRINT fs$(p%)
 2410 MOVE fx%+eye%,fy%:PRINT eye$
 2420 PROCbox(fx%,fy%)
 2430 ENDPROC
 2440 DEFPROCman
 2450 FOR pos%=1 TO 2
 2460 REPEAT:VDU4,17,0
 2470 PRINTTAB(1,3);"Enter ";
 2480 PRINT;n$(pos%);" choice"
 2490 REPEAT:y%=GET-64
 2500 UNTIL y%>0 AND y%<9
 2510 REPEAT:x%=GET-48
 2520 UNTIL x%>-1 AND x%<5
 2530 X%=x%+(y%-1)*4
 2540 fx%=x%*320-224:fy%=1064-y%*96
 2550 UNTIL tally%(X%)=0 AND x%<>0
 2560 PROCprfish
 2570 NEXT
 2580 ENDPROC
 2590 DEFPROCcomp
 2600 FOR J%=1 TO 6
 2610 IF tally%(cm%(J%))=1 cm%(J%)=0
 2620 NEXT
 2630 ff%=0
 2640 PROCtmf(5,4)
 2650 IF ff%>0 pos%=1:X%=cm%(5):PROCff:p
os%=2:X%=cm%(ff%):PROCff:ENDPROC
 2660 IF ff%=0 THEN PROCtmf(6,4)
 2670 IF ff%>0 pos%=1:X%=cm%(6):PROCff:p
os%=2:X%=cm%(ff%):PROCff:ENDPROC
 2680 IF ff%=0 THEN PROCrnd:cm%(0)=X%:ta
lly%(X%)=1:PROCtmf(0,6)
 2690 IF ff%>0 pos%=1::PROCff:pos%=2:X%=
cm%(ff%):PROCff:ENDPROC
 2700 IF ff%=0 THEN tally%(X%)=0:pos%=1:
PROCff:PROCrnd:pos%=2:PROCff
 2710 ENDPROC
 2720 DEFPROCtmf(A%,B%)
 2730 FOR J%=1 TO B%
 2740 IF fish%(cm%(A%))=fish%(cm%(J%)) A
ND cm%(A%)<>cm%(J%) THEN ff%=J%
 2750 NEXT
 2760 ENDPROC
 2770 DEFPROCrnd
 2780 REPEAT:VDU4,17,0
 2790 X%=RND(32)
 2800 UNTIL tally%(X%)=0
 2810 ENDPROC
 2820 DEFPROCff
 2830 x%=(X%-1) MOD 4+1
 2840 y%=(X%-1) DIV 4+1
 2850 X%=x%+(y%-1)*4
 2860 fx%=x%*320-224:fy%=1064-y%*96
 2870 PROCprfish
 2880 ENDPROC
 2890 DEFPROCrub
 2900 FOR pos%=1 TO 2
 2910 GCOL0,0
 2920 fx%=xp%(pos%)*320-224
 2930 fy%=1064-yp%(pos%)*96
 2940 PROCbox(fx%,fy%)
 2950 X%=xp%(pos%)+(yp%(pos%)-1)*4
 2960 MOVE fx%,fy%:PRINT f$(0)
 2970 GCOL0,2
 2980 a$=CHR$(yp%(pos%)+64)+CHR$(xp%(pos
%)+48)
 2990 MOVE fx%,fy%:PRINT a$
 3000 PROCupdate(X%)
 3010 tally%(X%)=0
 3020 NEXT
 3030 py%=py% MOD 2+1
 3040 ENDPROC
 3050 DEFPROChold
 3060 FOR pos%=1 TO 2
 3070 PROCbox(xp%(pos%)*320-224,1064-yp%
(pos%)*96)
 3080 NEXT
 3090 n%=n%+1
 3100 fco(py%)=fco(py%)+2
 3110 IF bt%=6 bonus%(py%)=bonus%(py%)+1
00
 3120 ENDPROC
 3130 DEFPROCbox(l%,u%)
 3140 GCOL3,3:MOVEl%-88,u%+32
 3150 PLOT17,0,-92:PLOT17,288,0
 3160 PLOT17,0,92:PLOT17,-288,0
 3170 GCOL0,0
 3180 ENDPROC
 3190 DEFPROCupdate(M%)
 3200 FOR J%=0 TO 5:cm%(J%)=cm%(J%+1):NE
```

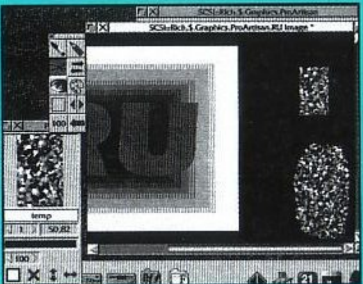
```
XT
 3210 cm%(6)=M%
 3220 ENDPROC
 3230 DEFPROCtable
 3240 SOUND1,2,100,20
 3250 FOR py%=1 TO 2
 3260 IF pl$(2)="Mike Roe" THEN time%(2)
=time%(2)+5000
 3270 tb%=300-time%(py%)DIV 100
 3280 IF tb%<0 tb%=0
 3290 ft%=(fco(py%)/turn(py%))*1000
 3300 pts%(py%)=ft%+tb%+bonus%(py%)
 3310 NEXT
 3320 VDU19,0,4;0;19,1,3;0;
 3330 FOR py%=1 TO 2
 3340 PRINTTAB(py%*15-2,2);pl$(py%)
 3350 PRINTTAB(py%*15,4);bonus%(py%)
 3360 PRINTTAB(py%*15,6);fco(py%)
 3370 PRINTTAB(py%*15,8);turn(py%)
 3380 PRINTTAB(py%*15,10);time%(py%) DIV
 100
 3390 PRINTTAB(py%*15,13);pts%(py%)
 3400 NEXT
 3410 PRINTTAB(4,4);"bonus"
 3420 PRINTTAB(5,6);"fish"
 3430 PRINTTAB(4,8);"turns"
 3440 PRINTTAB(0,10);"time(secs)"
 3450 PRINTTAB(3,13);"SCORED"
 3460 MOVE 0,500:PLOT 21,1279,500
 3470 PRINTTAB(11,17)"Top anglers table"
 3480 FOR j%=1 TO 4
 3490 PRINTTAB(10,j%*2+17);tab$(j%),tab%
(j%)
 3500 NEXT
 3510 a=INKEY(800)
 3520 FOR k%=1 TO 4:FOR j%=10 TO 39
 3530 PRINTTAB(j%,k%*2+17);" "
 3540 NEXT:NEXT
 3550 FOR py%=1 TO 2:pos%=1
 3560 FOR j%=1 TO 4
 3570 IF tab%(j%)>=pts%(py%) pos%=pos%+1
 3580 NEXT j%
 3590 FOR j%=4 TO pos% STEP-1
 3600 tab$(j%+1)=tab$(j%)
 3610 tab%(j%+1)=tab%(j%)
 3620 NEXT
 3630 tab$(pos%)=pl$(py%)
 3640 tab%(pos%)=pts%(py%)
 3650 NEXT
 3660 FOR j%=1 TO 4
 3670 PRINTTAB(10,j%*2+17);tab$(j%),tab%
(j%)
 3680 NEXT
 3690 ENDPROC
 3700 DEFPROCscore
 3710 VDU4,12,17,2-py%
 3720 PRINTTAB(1,0)pl$(py%);"'s turn"
 3730 PRINTTAB(1,5);"Fish  ";fco(py%);
 3740 PRINTTAB(12,5);"turn ";turn(py%)
 3750 VDU5
 3760 ENDPROC
 3770 DATA133,133,133,133,10,8,8,8,8
 3780 DATA133,133,133,133,10,8,8,8,8,133
,133,133,133
 3790 DATA128,129,130,131,10,8,8,8,8,132
,133,134,135
 3800 DATA10,8,8,8,8,136,137,138,139
 3810 DATA140,141,142,143,10,8,8,8,8,144
,145,146,147
 3820 DATA10,8,8,8,8,148,149,150,151
 3830 DATA152,153,154,155,10,8,8,8,8,156
,157,158,155
 3840 DATA10,8,8,8,8,159,160,161,155
 3850 DATA162,163,164,165,10,8,8,8,8,155
,166,167,168
 3860 DATA10,8,8,8,8,169,170,171,172
 3870 DATA173,174,175,176,10,8,8,8,8,155
,177,178,179
 3880 DATA10,8,8,8,8,180,181,182,183
 3890 DATA 1,0,1,-1,3,0,2
 3900 DATA 1,0,1,0
 3910 DATA 1,0,2,-1,2,0,3
 3920 DATA 1,0,3,-1,3,0,1
 3930 DATA 4,0,2,-1,5,0,1
 3940 DATA 4,0,2,0
 3950 DATA 4,15,1,-1,5,0,1
 3960 DATA 5,0,1,-1,4,19,2
 3970 DATA 18,0,3,184,18,0,0,8,185
 3980 DATA"  Ali Butt",1000,"Len M.Sole"
,900
 3990 DATA"Ivor Trout",800,"Roly Skate",
700
```

RISC user

The number one subscription magazine for the Archimedes

RISC User, probably the most popular subscription magazine for the Archimedes, offers all the information you need as an Archimedes user. In every issue of RISC User you will find a wealth of articles and programs with professionally written reviews, lively news, help and advice for beginners and experienced users, and items of home entertainment.

The B5 size of RISC User allows a sophisticated design, big colour illustrations and pages full of information, and yet is still a convenient size to assemble into an easy-to-use reference library. Altogether, in its six years of existence, RISC User has established a reputation for a professional magazine with accurate, objective and informed articles of real practical use to all users of Acorn's range of RISC computers.



SUBSCRIPTION DETAILS

As a member of BEEBUG you may subscribe to RISC User for the reduced rate of £18.40 (a saving of £1.50 on the normal subscription rate).
Overseas subscription rates are as follows:
£27.50 Europe and Eire, £33.50 Middle East, £36.50 Americas & Africa, £39.50 Elsewhere

Contents of the latest Vol.7 Issue 4 of RISC User:

DOTS BEFORE YOUR EYES
A way of creating 3D images which can be viewed without any artificial aids.

WHAT'S COOKING AT COLTON?
An insider's view of Colton Software's philosophy from Mark Colton himself.

NEW SPRITES
An examination of Acorn's future new sprite formats for 16-bit and 24-bit colour.

MORPHOLOGY
A review of two packages providing contrasting capabilities in the field of morphing - the 'in' topic of the moment.

CANON BJC-600 COLOUR PRINTER
A review of one of the most cost effective colour printers yet to appear.

AUDIOWORKS & WORDWORKS
A review of two new but quite different packages from Computer Concepts, one for text one for sound.

KEY MOVES
A survey of the extensive range of software and supporting resources for schools produced by Anglia TV.

WRITE-BACK
The readers' section of RISC User for comment, help, information - a magazine version of a bulletin board.

INTO THE ARC
A regular series for beginners.

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

COLOURED VIEW DISPLAY

S.Fagg

The usual way to change the colour of the display in modes other than 7 would be to use the VDU command, however VIEW does not support this. To get at the colours you must use the control characters more directly e.g.

```
*KEY0,|S|A|B|@|@|@|M etc.
```

This will give green text on a black background when f0 is pressed in the command mode. While in text mode the function keys are disabled, they may be re-enabled with *FX225,1.

SAFER ESCAPE ACTION

Safer ways of obtaining an Escape condition through the use of *FX220 are:

*FX220,0	= Ctrl-@ produce Escape
*FX220,128	= f0 produce Escape
*FX220,144	= Shift-f0 produce Escape
*FX220,160	= Ctrl-f0 produce Escape
*FX220,176	= Shift-Ctrl-f0 produce Escape

The last command is an extremely useful and safe 'three-key' Escape action.

COLOURED TITLES

This procedure enables you to highlight a disc title by using Teletext codes. Whilst coloured filenames are not possible on the disc, the disc title may be made coloured to make it stand out when a *CAT is done. Press Shift plus the function key for the appropriate teletext graphics character, followed by the text.

PREVENTING THE SCREEN FROM SCROLLING

P.Davies

The screen will automatically scroll if a character is printed in the bottom right corner of the screen. This will have the effect of obscuring any text on the top line. An easy way to prevent this is to type in ?&D0=2. This sets the second bit of the location &D0, which controls the scrolling facility, so that it is inhibited. After printing a character in this position, the location must be returned to its former state by entering ?&D0=0.

OPENOUT BUG

Writing to an existing random access file that is write protected will generate an error but will also flush the buffer resulting in the first 256 characters of the output data in memory being lost.

SCREEN AND WINDOW WIDTH

To calculate the width, in characters, of the screen or a text window regardless of the display mode the following short routine can be used:

```
VDU 13,8
width=POS+1
```

PERSONALIZED HEADER ON BREAK

There are more useful ways to use the Break vector, but the short piece of code below will personalize your title banner on Break.

```
  10 osasci=&FFEE3:osbyte=&FFF4:PROCasse
mble:CALL init:END
  20 DEFPROCassemble
  30 FORpass%=0 TO 3 STEP 3:P%=&C00
  40 [OPT pass%:.start BCC exit:LDX#11:L
DY #0
  50 .print LDAmess,Y:JSRosasci:INY:DEX:
BN Eprint:.exit RTS
  60 .init LDY#0:LDA#&F7:LDX#&4C:JSRosby
te:LDA#&F8:LDX#start MOD 256:JSRosbyte
  70 LDA#&F9:LDX#start DIV 256:JSRosbyte
:RT S:.mess:]
  80 $P%="Your name":P%=P%+9
  90 ?P%=13:P%?1=13:P%=P%+2
 100 NEXT:ENDPROC
```

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 all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS.

Cassette based games and utilities too many to list please phone for list, also BBC B with DFS but Analogue port seems u/s £50, some games and applications on both disc and tape, a bit of 'pot-luck' £20, single sided 5.25" 40/80T drive (bare, bur hopefully dad is making a box for it) £15, ATPL board + two 8Kb RAM chips £25, Vine 'Replay' for 8721 controller £25, also a large number of integrated circuits of various types, mostly 74 series and similar 'pot-luck' for a job lot say £2 plus postage. Tel. Nr Pontypridd 0443 206 771.

M128 fitted with 512 co-processor complete with Gem mouse and software, BEEBUG Exmon II and Master ROMs, Replay ROM, twin 40/80T disc drives with own PSU, manuals including View, Viewsheet Master Reference I & II software on disc and tapes £300 o.n.o. Tel. Brighton 0273 415892.

WANTED: M128, computer alone or complete system, also a copy of Viewsheet User Guide. Tel. Cambridge 0223 891806.

M128, Cumana single 40/80T drive, Panasonic KX-P1081 printer, all in excellent condition, manuals and books inc. reference manuals, complete 5 years BEEBUG magazines, many 5.25" discs £250. Tel. Bucks 0628 521231.

BEEBUG magazines from vol. 5/8 to 6/4 inclusive £4, Econet module and ANFS ROM for Master £10, ADFS ROM £8, Edword W/P ROM £4, Clares Replica £3, Acorn Data Recorder £10, various popular games on tape £1 each, BBC User Guide as new £3. Tel. Perth 0738 812186.

Tandata Td1400 Viewdata Terminal, 1200/75bps with BBC terminal software, little used perfect condition with manuals. Offers Tel. Cheshire 0928 722454.

DTP package "Wapping Editor", comprising 1 No ROM, Art disc, Utilities disc and Operating manual, complete with 'Quest' mouse £50. Tel. Kent 0732 863105.

Oki dot matrix printer with tractor feed, brand new printer never been used £50, BBC B plus complete with View £20, Microvitec Cub colour monitor £25, Philips b/w monitor £15, Opus Challenger (never used) £20, double disc drive in plinth £20, double disc drive in block £20, Solidisc RAM/ROM £20, Juki daisywheel printer £20. Tel. Ipswich 0473 311107.

WANTED: Beeb DOS software to convert files written in View to a suitable format for Windows. Tel. Herts 0442 833263.

WANTED: Pres AP5 and/or User Port cartridge for Electron, good prices paid. Tel. Edinburgh 031-225 5563.

BBC B issue 7 DFS £55, Akhter bridged dual disc drives 40/80 £50, JVC colour monitor for BBC B/Master £50, Sleuth EPROM £7, Exmon II EPROM £7, Acornsoft Basic Editor ROM £7, Watford Buffer/Backup ROM £5, original software on tape and disc, books, magazines etc. must clear. Send for list. Tel. Gwynedd 0286 880997.

WANTED: Master ROM, Spellmaster, Wapping Editor DTP, Printmaster 1770 DFS. Tel. 0582 413990.

BEEBUG Vol. 1.1 to date, Acorn User most of No.1 to 29, The Micro User 1.1 to 4.1. Tel. Telford 0952 812644.

WANTED: Flight simulators for BBC B on 5.25" discs, Aviator 747, 737, 767 etc. no shoot em-ups. Tel. Bristol 0454 417809.

BBC B + 128 sideways RAM, dual double sided 40/80T disc, philips monitor, joystick, software including Wordwise Plus, Spellmaster, Diagram II with set of manuals and selection of games. Offers please Tel. Ledbury 0531 633470.

BBC B early model with Integra Beta o/s, 1770 DFS and ADFS, 2nd processor with expansion up to 1Mb, CMC 20Mb Winchester, Opus 40/80T 5.25" double disc drive with PSU, Zenith mono monitor, Star Gemini printer, Z80 2nd processor, CC Mega 3 Spellmaster, Interbase, DOS Wordprocessor database spreadsheet Gem software plus plenty of Shareware. To be purchased as single item £275, buyer collects. Mr Hardy, Linden, Lime Walk, Dibden Purlieu, Southampton SO4 5RA. Tel. 0703 842646.

M128 Turbo, Acorn Cambridge 1Mb 32bit 2nd processor, Microvitec monitor, 56Mb hard disc, dual 40/80T drives, Pace modem, entire Inter and View families, C, Pascal, Forth, BCPL, Fortran, Lisp, Wordwise Plus with many manuals, books, magazines etc. real bargain £750 the lot (plus carriage). Tel. Nottingham 0602 421498.

CORPLAN

- The proven document management system for all versions of Wordwise Plus and Wordwise Plus 2 in Model B, B+, Master and Compact. Reviewed in Beebug Vol. 9 No. 8.
- Unique interactive system makes professional-style letters and documents easy to produce and easy to manage.
- Twin disc drives are recommended, but in ADFS and certain enhanced DFS variants, a single drive can suffice.
- Supplied on installation disc (read-only 40T DFS) with comprehensive manual. Order other format if needed.
- Price to Beebug members £17.50 UK post paid. Cheques payable to C W Robertson, please. 14-day unconditional refund. Free information leaflet available.

Colin W Robertson
Three Gables, 7a Talbots Drive, Maidenhead, Berks, SL6 4LZ
Phone/fax (0628) 24591

ENHANCED HEARING

Having typed in and used the 'Hearing Test' program (BEEBUG Vol.12 No.4), it occurs to me that it would be easier to let the computer record the levels as they are decided, and then print them out so that they could be entered onto graph paper at one's leisure. In this way one can concentrate on the test without having to bother with recording the results. I have found this can be achieved by inserting the following lines into the program:

```
  60  DIM store(100)
  65  DIM freq(100)
  70  L=1
  80  tone=131
1382 store(L)=yournoise%
1384 freq(L)=tone
1386 L=L+1
1388 tone=tone*2^(1/12)
1455 LOCAL P
1465 CLS: FOR P=1 TO L-1
1467 VDU 2
1470 PRINT "  "; P "  Level ";store(P);
" at a frequency of ";INT(freq(P))
1475 NEXT P: VDU 3
1477 ENDPROC
1480 :
```

B.W.Fursman

BACK TO BASICS REVISITED

I was most interested in Mr.Ambrose's letter in *Postbag* (BEEBUG Vol.12 No.7), as only a few days ago, out of curiosity, I borrowed and digested a neighbour's manual for the GW-Basic bundled with a recently purchased 386 PC, and was struck by how poor it seemed in comparison with Basic IV on my Master Compact. And there is the still more advanced Basic VI for the Archimedes, with which I am not familiar. I suggest that Mr.Ambrose do the same, and peruse a GW-Basic manual, when all his questions will be answered.

Mr.Ambrose, I am sure, is right not to change lightly from his old Master. If you want a Windows-like interface, and the possibility of using commercially produced software for DTP, advanced graphics, the ultimate in speed etc, of course you need a more modern machine with millions of bytes. If, however, your interest is in programs of your own devising, 32K of RAM (64K on the Master series with Basic 128) should, I think, be enough. Programs that are on the face of it too long can be split up and chained. Whether it is worth upgrading to an Archimedes just for the sake of Basic VI is something that Mr.Ambrose could decide by reading the Archimedes Basic manual.

Ruben Hadekel

SOME BASIC HISTORY

You might like to refer Mr.Ambrose (Postbag, Vol.12 No.7) to the article by Bill Gates published in *Byte* October 1989 entitled *25th Birthday of BASIC"*, and the book by Kemeny & Kurtz *Back to Basic* published by Addison-Wesley in 1985. The starting date for students at Dartmouth College using Basic is given as May 1964. Kurtz was maths professor, and Kemeny chairman of the maths department. The idea was that *all* students at Dartmouth should learn about computers. The only way to achieve this was to develop a timesharing operating system and develop a simpler language than Algol or Fortran. One of Gates' own claims to fame is his and Allen's Basic interpreter for the Altair 8080 with 4K of memory in 1974/75. There is an account of this in *Hard Drive* by J.Wallace and J.Erikson (a biography of Bill Gates) published by John Wiley. This was serialised in Personal Computer World from August 1992 onwards.

John Sutton

Any letters for possible publication in the last issue of BEEBUG should be sent in as quickly as possible.

Extended Keyboard (continued from page 11)

came with your Master, including the exact space - <highlight 2>@ (*@).

You should note that View does not expect special characters on its command screen (they are converted into full stops, in fact), so searching for special characters must be done using the wildcard sequence ^?. The READ and WRITE commands strip our special characters out, so if you need to merge files, paste them together in Edit first.

MAKING A WORKDISC

All you actually need when using AltRom for your work is a copy of the ROM image AltRom itself and the printer driver for View, Epson. The other files are no longer needed (but keep them safe!). Only KeyCaps needs the data files to be in the current directory when it runs.

I recommend that you copy AltRom and Epson to your workdisc and set up a boot file to load and initialise the ROM as follows - you may want to change details:

```
*| >!Boot for AltKey
*BASIC
*IBM
*KEYB UK
*SRLOAD AltRom 8000 4Q
?&2A5=&82
*KEY 10 *WORD|MPrinter Epson|M
CALL !-4
```

This procedure simulates pressing Break to initialise the ROM image. It is not necessary to use Ctrl-Break if you do this. The definition of Key 10 should contain whatever you want. My own disc has:

```
*Key 10 "*Word|MSet FI|MPrinter
Epson|MMode 3|M*.|MLoad "
```

but you may wish to enter a disc menu program or something else.

NEXT MONTH

In the next issue we will look at defining your own layouts and getting the best out of AltRom.

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

Last month's article in the *Wordwise User's Notebook* series was incorrectly attributed to Colin Robertson. The author was in fact Chris Robbins, and we are sorry for not having given him the credit he deserves at the time.

SPECIAL OFFERS FOR BEEBUG READERS

As publication of BEEBUG approaches its final issue next month, we will be selling all remaining stocks of BEEBUG magazines and magazine discs, plus our Best of BEEBUG range at special prices. A separate leaflet included with this issue covers magazines and magazine discs. Stocks are limited in all cases, and current availability is shown on the leaflet. This may change so it may be better to check by phone when placing your order.

Best of BEEBUG discs, advertised in our centre pages, are also available at special prices in most cases. The same comments about availability apply here too. Please note that *Magscan* is now unavailable in any format. However, we have uncovered a small quantity of *BEEBUG Filer*, a database management system originally published in the magazine, a disc which subsequently proved to be very popular in an updated stand-alone form. This provides all basic data handling functions including searching and sorting, comprehensive report generation, plus subsidiary programs for home accounts and graphs.

We will be repeating these offers, subject to availability, in the April issue of BEEBUG, due out late March. No further orders for BEEBUG magazine and associated products will be accepted after 30th April 1994.

To order, write to
BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS, or tel. 0727 840303, fax 0727 860263.

Magazine Disc

March 1994

EXTENDED KEYBOARD - The collection of programs and other files providing extended keyboard capabilities for the Master 128.

FISH - This is an unusual implementation of pelmanism which requires you to match up pairs of strange and colourful fish.

DUAL DUMP - A handy utility which enables you to compare the contents of two files displayed side by side on the screen.

CUTE KEY CUTS - Another useful utility of particular interest to Wordwise (Plus) users, for defining function key operations.

BODY BUILDING - A striking example of the Beeb's graphics capabilities enables you to construct lifelike humanoids.

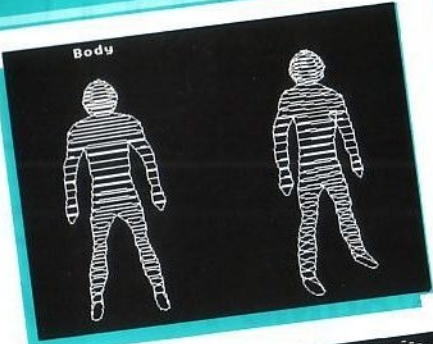
GRAFFER - This disc contains the additional programs to work alongside Graffer, the graph plotting program published last month.

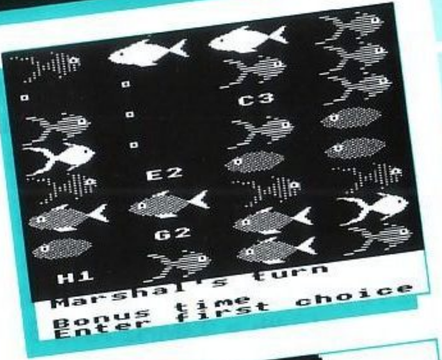
BEEBUG WORKSHOP - Two sets of functions and procedures for date handling, as described in the magazine, and a separate demonstration of the use of these dating routines.

MAGSCAN DATA - Bibliography for this issue of BEEBUG (Vol.12 No.9).

BONUS ITEMS

FRUIT MACHINE - A completely new and previously unpublished implementation of a fruit machine - just the way to fill all those idle moments.





ALL THIS FOR £4.75 (5.25" & 3.5" DISC) + £1 P&P (50p FOR EACH ADDITIONAL ITEM)
Back issues (5.25" and 3.5" discs from Vol.6 No.1) available at the same prices.
Prices are inclusive of VAT and postage as applicable. Sterling only please.

RISC Developments, 117 Hatfield Road, St.Albans, Herts AL1 4JS

STOP PRESS . . . STOP PRESS . . . STOP PRESS . . . STOP PRESS . . . STOP PRESS . . .

BEEBUG

Don't miss out - ORDER NOW!

A5000 for only £850

2Mb/80Mb HD LEARNING CURVE

Once again we have managed to obtain a number of A5000 Learning Curve systems at a ridiculously low price. Only a few are left in stock, so hurry if you are not to miss out.

The A5000 on offer is the top of the range computer from Acorn, and features the 25MHz ARM3 processor (the latest models are equipped with a 33MHz ARM3 making them marginally faster).

We are offering a range of specially priced upgrades provided they are all supplied or fitted at the same time of your order for the A5000. All are fully guaranteed for 12 months along with the computer. So whether you can only afford the computer now at £850, or a full blown system we can supply exactly what you need.

A5000 Specification

RISC OS 3.1
2Mb RAM
80Mb Hard drive
25MHz ARM3

Learning Curve pack

PC Emulator v1.7
DOS 5
First Word Plus
Acorn DTP
Audio Training Tape
Pacmania
Genesis Plus

Additional Hard Drives:

160Mb	£199
260Mb	£279
450Mb	£449

These drives are fitted in addition to the standard 80Mb and do not replace them.

Monitor options:

Standard (AKF40)	£149
MultiScan (AKF52)	£179
High Res. (AKF50)	£250

RAM Upgrades:

2-4Mb	£94
2-8Mb	£379

BEEBUG Ltd 117 Hatfield Road, St Albans, Hertfordshire AL1 4JS.
Tel: 0727 840303 Telesales Direct: 0727 840305 Fax: 0727 860263
Prices shown are exclusive of £8.00 for carriage and VAT.