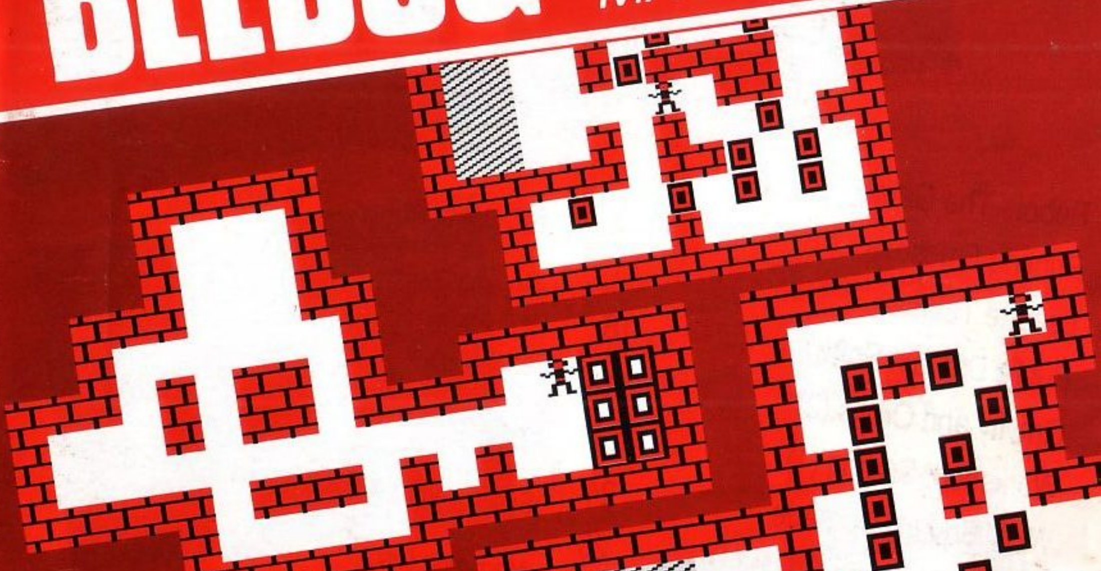
Vol.11 No.10 April 1993

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



Robol - The Game

- PRINTING FRACTIONS
- SORTING
- DATASHEET UPDATED
- LEAGUE TABLES

BEEBUG

Vol.11 No.10 April 1993

FEATURES

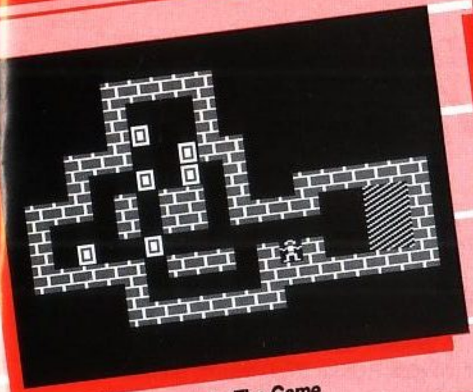
REGULAR ITEMS

HINTS & TIPS

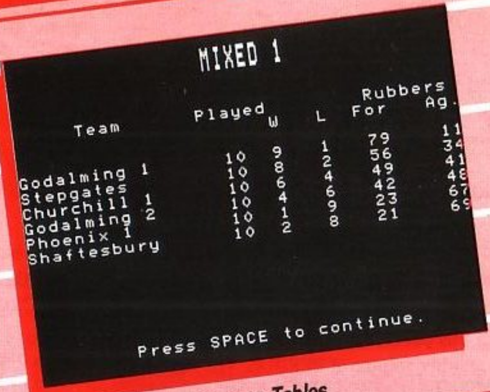
PROGRAM INFORMATION

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

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

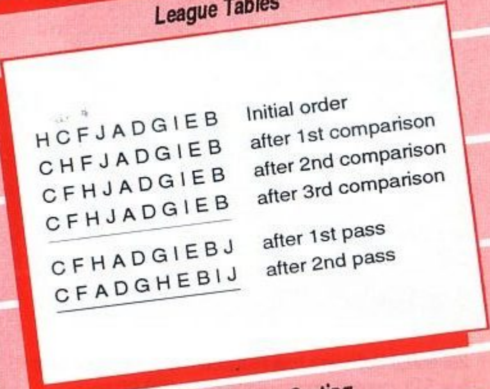


Robol - The Game



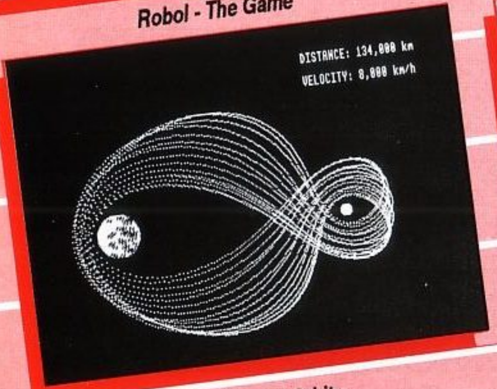
```
                     MIXED 1
                                        Rubbers
                   Played          L   For   Ag.
       Team               W
                                        79     11
                     10   9    1        56     34
                     10   8    2        49     41
Godalming 1          10   6    4        42     48
Stepgates            10   4    6        23     67
Churchill 1          10   1    9        21     69
Godalming 2          10   2    8
Phoenix 1
Shaftesbury

          Press SPACE to continue.
```

League Tables

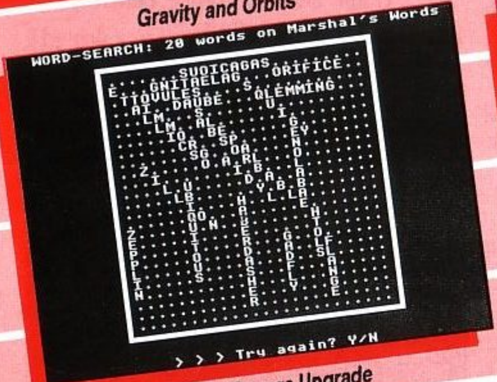


HCFJADGIEB	Initial order
CHFJADGIEB	after 1st comparison
CFHJADGIEB	after 2nd comparison
CFHJADGIEB	after 3rd comparison
CFHADGIEBJ	after 1st pass
CFADGHEBIJ	after 2nd pass

Workshop: Sorting

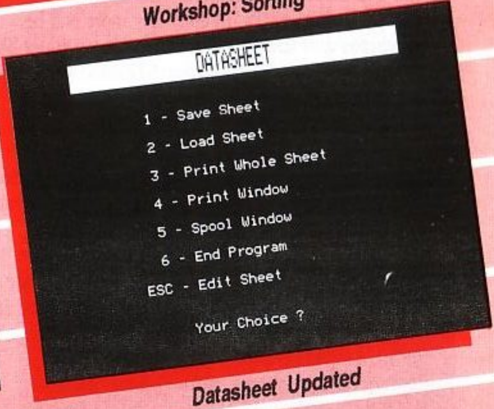


Gravity and Orbits



```
WORD-SEARCH: 20 words on Marshal's Words

> > > Try again? Y/N
```

Word Square Upgrade



```
DATASHEET

1 - Save Sheet
2 - Load Sheet
3 - Print Whole Sheet
4 - Print Window
5 - Spool Window
6 - End Program
ESC - Edit Sheet

Your Choice ?
```

Datasheet Updated

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

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

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings/News

Volume 11 Index

With this particular issue we come to the end of yet another volume of BEEBUG. The next issue will see the start of our twelfth year of publication. As is our custom we shall be preparing a complete printed index to the whole of volume 11, and this will be distributed to all subscribers with Vol.12 No.1. In addition, indexes to previous volumes of BEEBUG can normally be supplied on request provided that you send us a stamped addressed envelope for this purpose. Please send any such requests to BEEBUG Magazine, at our usual St.Albans address.

Beeb Bargains

Essential Software has reduced the price of its last few 512 memory expansions from £99.00 to £75.00 inclusive. It's unlikely any more will be produced and less than ten remain, so if you'd like to expand the memory of your 512 don't put it off any longer. Enquiries to: Essential Software, PO Box 5, Groby, Leicester LE6 0ZB.

Dabs Press has given permission for Robin Burton to supply a revised, 24-pin version of HyperDriver, his Epson compatible printer driver (reviewed BEEBUG Vol.6 No.9, summary Vol.7 No.7). The upgrade price is £10.00 for EPROM, or £8.00 for the disc version, both inclusive. You must send an original EPROM or an issue disc, making cheques payable to R.D.Burton, or send an S.A.E. for further details to Robin at the above address.

BladeDancer

We can report this month on a brand new game for the BBC micro range. *BladeDancer* from Omicron Technologies is a graphics adventure game in which you choose the character you wish to play (Archer, Wizard or Warrior) in solving over 150 puzzles in 650 different locations. You must find the nine pieces of the Pentagram and banish the Demon to its own plain before it destroys the Norse god of war, Galac Thaar.

BladeDancer is supplied on five 5.25" discs for use with the BBC micro or Master 128, and costs £11.95 inclusive direct from Omicron Technologies, P.O.Box 37, Daventry, Northants NN11 4UG.

All Formats Computer Fairs

If you haven't used up the free ticket which we distributed last month with each issue of BEEBUG, then here is the next selection of All Formats Computer Fairs to choose from.

3 Apr	Adam House, Chambers Street, Edinburgh.
4 Apr	City Hall, Candleriggs, Glasgow.
11 Apr	Corn Exchange, Church Street, Brighton.
17 Apr	Jesse Boot Centre, University of Nottingham.
18 Apr	National Motorcycle Museum, NEC, Birmingham (J6 M42).
24 Apr	Sandown Park, Esher, Surrey (J9/10 M25).
25 Apr	Brunel Centre, Temple Meads, Bristol.
1 May	Northumbria Centre, Washington, County Durham.
2 May	University Sports Centre, Calverley Street, Leeds.

All fairs run from 10am to 4pm. Telephone 0608 663820 for more information.

Spring in Harrogate

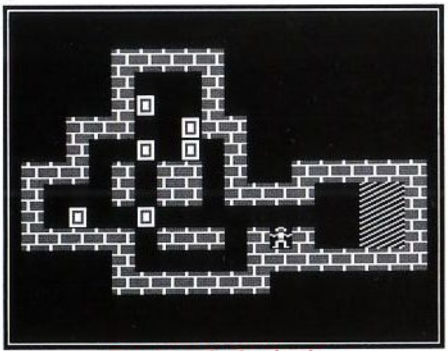
The BBC Acorn User Show takes place in Harrogate from 15th to 17th April at the Harrogate International Centre. RISC Developments will have a stand there where we look forward to meeting BEEBUG readers.

M.W.

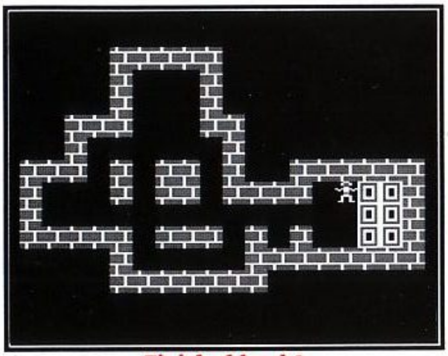
Robol - The Game

Miroslaw Bobrowski shows that shifting boxes can be fun.

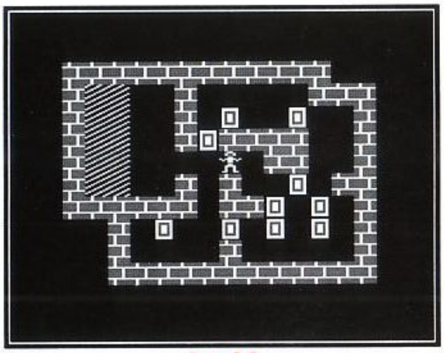
'Robol' is a pejorative expression used in the Polish language for a dullish labourer and as such it is used in the same context as the English word drudge. 'Robol' is also a hero of a multilevel strategy game, which is, in fact, a BBC-Micro version of the IBM game called *Sokoban*.


Start at the beginning

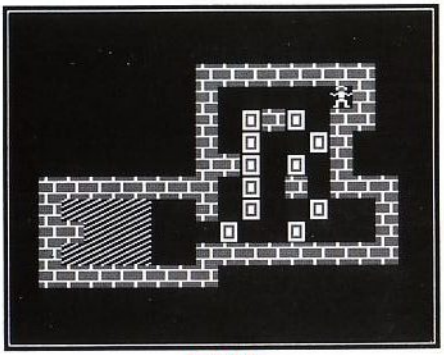
The rules of the game are very simple; all you have to do is to move the Robol character round the screen to push all the


Finished level 1

packs onto the target shown as a cross-hatched area. The first levels are very easy to complete within a few minutes but you need more than 20 minutes to find a solution when playing higher levels. Furthermore, one reckless move can make it impossible to complete a level.


Level 2

The Robol program consists of several files. The first one, *Robol* (Listing 1), contains a short instruction screen, sprite data, and two pieces of code for assembling. The second file, *Robol2* (Listing2), is the main program controlling the game. The third file, *RobData* (Listing 3), generates the robol3 data file, which is called by Robol2.


Level 3

There can be fifty screen levels ready for use and these are stored in a compact form between locations &3000 and &4F3F (a space of &A0 bytes is required for each screen). Theoretically there is a space for 14 additional screens (from &4F40 to &57FF).

Because of the space it would take up, the printed listing of RobData contains only ten screens. The number of screens is controlled in the lines 120, 130, 2520 and 2530 in RobData and in the 1610 in Robol2. The version on the disc has the full 50 screens so those lines will be different.

When all three listings have been typed in, fully debugged and saved on the disc, then run RobData to produce the robol3 file. To run the game enter CHAIN "ROBOL". The other files are then loaded automatically, and you can start to play the game at level 1. To change a level press the 'L' key and enter any number that is not greater than 10 (50 on the disc version). To move the Robol character round the screen and to push the packs use the 'Z','X','/' or ':' keys. The Escape key should be used to restart the game at the current level, whereas pressing Ctrl-Escape enables you to exit the program. Next month I'll be presenting a screen designer to let you create your own levels. Now, get shifting!

```
   10 REM Program ROBOL
   20 REM Version B 2.1
   30 REM Author  M.Bobrowski
   40 REM BEEBUG  April 1993
   50 REM Program subject to copyright
   60 :
  100 MODE 7:VDU 23;8202;0;0;0;
  110 ENVELOPE 1,4,-1,1,-1,1,1,1,20,10,0
,-30,80,100
  120 PROCspritedata:PROCdisplay_spr:PRO
Ctune
  130 PROCinstr
  140 :
  150 CHAIN"Robol2"
  160 :
 1000 DEF PROCspritedata
 1010 P%=&C00
 1020 [OPT 0
 1030 EQUS STRING$(32,CHR$0)
 1040 ]
 1050 FOR I%=1 TO 192 STEP 4
 1060 READ a$
 1070 [OPT 0
 1080 EQUD EVAL("&"+a$)
 1090 ]
 1100 NEXT
 1110 ENDPROC
 1120 :
 1130 DATA FF2F2F2F,F0F0F0F,FF0F0F0F,2F2
F2F2F,FF0F0F0F,4F4F4F4F,FF4F4F4F,F0F0F0F
 1140 DATA 22118844,22118844,22118844,22
118844,22118844,22118844,22118844,221188
44
 1150 DATA 43707000,53535343,43535353,70
7043,2CE0E000,ACACAC2C,2CACACAC,E0E02C
 1160 DATA 1212110,1155BB11,20301001,602
020,8484880,88AADD88,40C08008,604040
 1170 DATA BC8F8FFF,ACACACBC,BCACACAC,FF
8F8FBC,D31F1FFF,535353D3,D3535353,FF1F1F
D3
 1180 DATA 1212110,1155BB11,20301001,602
020,8484880,88AADD88,40C08008,604040
 1190 :
 1200 DEF PROCdisplay_spr
 1210 scr_addr=&70:temp=&72:row=&74
 1220 temprow=&75:column=&76:index=&77
 1230 x=&78:y=&79:addr=&7A
 1240 FOR pass=0 TO 2 STEP 2
 1250 P%=&B00
 1260 [OPT pass
 1270 STX x:STY y
 1280 ASL A:ASL A:ASL A:ASL A:ASL A:STA
index
 1290 LDA #&58:STA addr+1:LDA #0:STA add
r
 1300 LDY y:BEQ zero
 1310 .loop1
 1320 CLC:LDA addr:ADC #&40:STA addr
 1330 LDA addr+1:ADC #1:STA addr+1
 1340 DEY:BNE loop1
 1350 .zero
 1360 LDX x:BEQ store_addr
 1370 .loop2
 1380 CLC:LDA addr:ADC #16:STA addr
 1390 LDA addr+1:ADC #0:STA addr+1
 1400 DEX:BNE loop2
 1410 :
 1420 .store_addr
 1430 LDA addr:STA scr_addr:LDA addr+1:S
TA scr_addr+1
 1440 LDA index:STA spr_addr+1:LDA #&0C:
STA spr_addr+2
 1450 :
 1460 LDX #2:LDY #16:STX column:STY row
 1470 LDX #0:LDY #0
 1480 LDA scr_addr:STA temp:LDA scr_addr
+1:STA temp+1
 1490 .display
 1500 LDA row:STA temprow
 1510 .spr_addr
 1520 LDA &FFFF,X:STA (scr_addr),Y
 1530 INX:LDA scr_addr:AND #7
 1540 CMP #7:BEQ bottom
 1550 INC scr_addr:BNE nextrow
 1560 INC scr_addr+1:JMP nextrow
 1570 .bottom
 1580 LDA scr_addr:ADC #&38:STA scr_addr
 1590 LDA scr_addr+1:ADC #1:STA scr_addr
+1
 1600 .nextrow
 1610 DEC temprow:BNE spr_addr
 1620 LDA temp:ADC #8:STA scr_addr:STA t
emp
 1630 LDA temp+1:ADC #0:STA scr_addr+1:S
TA temp+1
 1640 DEC column:BNE display
 1650 RTS
 1660 ]
 1670 NEXT
 1680 ENDPROC
 1690 :
 1700 DEF PROCtune
 1710 osbyte=&FFF4:osword=&FFF1
 1720 dptr=&70:work=&80
 1730 FOR pass=0 TO 2 STEP 2
 1740 P%=&900
 1750 [OPT pass
 1760 LDA #musdata MOD 256:STA dptr
 1770 LDA #musdata DIV 256:STA dptr+1
 1780 LDA #0:STA work+1:STA work+3
 1790 STA work+5:STA work+7
 1800 .music
 1810 LDA &FF:ROL A:BCC noesc
 1820 BCC noesc
 1830 LDA #&7E:JSR osbyte
 1840 BRK:EQUB 17:EQUS"Escape":BRK
 1850 .noesc
 1860 LDX #250:JSR adval
 1870 CPX #0:BEQ full
 1880 JSR sound1
 1890 .full
 1900 LDY #0:LDA (dptr),Y
 1910 CMP #&FF:BNE not_end
 1920 LDA #0:RTS
 1930 .not_end
 1940 JMP music
 1950 .adval
 1960 LDA #&80:LDY #0
 1970 JMP osbyte
 1980 .sound1
 1990 LDY #1:LDA (dptr),Y
 2000 CMP #&FF:BEQ out1
 2010 TAX:DEY
 2020 LDA (dptr),Y:TAY
 2030 LDA #1:JSR sound
 2040 CLC:LDA dptr:ADC #2:STA dptr
 2050 LDA dptr+1:ADC #0:STA dptr+1
 2060 .out1
 2070 RTS
 2080 .sound
 2090 STA work
 2100 STX work+6:STY work+4
 2110 LDA work+4:BNE norm
 2120 LDA #0:BEQ go
 2130 .norm
 2140 LDA #1
 2150 .go
 2160 STA work+2:ADC #0:STA work+2
 2170 LDX #work MOD 256:LDY #work DIV 25
6
 2180 LDA #7:JMP osword
 2190 .musdata
 2200 OPT FNdata
 2210 ]
 2220 NEXT
 2230 ENDPROC
 2240 :
 2250 DEF FNdata
 2260 RESTORE 2320
 2270 REPEAT
 2280 READ M%,N%:?P%=M%:P%?1=N%:P%=P%+2
 2290 UNTIL M%=255 AND N%=255
 2300 =pass
 2310 :
 2320 DATA 137,6,129,2,137,4,129,8,137,1
2,0,4,137,12,129,4,137,12,145,6,137,2,14
5,4,137,8,145,12,0,4,145,12,137,4,145,12
,0,4,157,12,151,4,157,12,0,4,141,12,133,
8,141,4,129,36,0,32,255,255
 2330 :
 2340 DEF PROCinstr
 2350 FORI%=1 TO 2:PRINTTAB(12,I%)CHR$15
7;CHR$131;CHR$129;CHR$141;"ROBOL";SPC4;C
HR$156:NEXT
 2360 PRINTTAB(12,24)CHR$130"Press SPACE
";:VDU28,0,22,39,4
 2370 PRINTCHR$134"Robol is a labourer e
mployed at the"'CHR$134"storehouse. Help
 him to push packs and"'CHR$134"to trans
port them to the cross-hatched"'CHR$134"
areas."
 2380 PRINTCHR$131"There are 50 game lev
els and you can"'CHR$131"choose any of t
hem during the game."
 2390 PRINTCHR$134"If no reasonable move
 is possible to"'CHR$134"complete a leve
l then press ESCAPE"'CHR$134"to start wi
th the same level again."
 2400 PRINT'SPC11;CHR$129"Control keys:"
''CHR$134;" Z ";CHR$131;"Move left";SPC7
;CHR$134;"X ";CHR$131;"Move right"'CHR$1
34;" * ";CHR$131;"Move up";SPC9;CHR$134;
"? ";CHR$131;"Move down"
 2410 PRINTCHR$134;" L ";CHR$131;"Change
level";SPC3;CHR$134;"ESC";CHR$131;"Resu
me game"''CHR$131;" To leave the game pr
ess";CHR$134"CTRL+ESCAPE";
 2420 REPEAT UNTIL GET=32:VDU26,12:ENDPR
OC
```

```
   10 REM Program Robol 2
   20 REM Version B 2.5
   30 REM Author  M.Bobrowski
   40 REM BEEBUG  April 1993
   50 REM Program subject to copyright
   60 :
  100 MODE 5:VDU 23;8202;0;0;0;:HIMEM=&3
000:*L.Robol3 3000
  110 DIM M% 320:*FX229,1
  120 L%=0:F%=FALSE:Z%=TRUE
  130 REPEAT:W%=0:N%=0:PROCscreen:IF Z%
THEN TIME=0
  140 REPEAT
  150 PROCgame
  160 UNTIL N%=0 OR F% OR G%
  170 IF N%=0 VDU 19,2,11;0;:CALL &900:L
%=L%+1:Z%=TRUE
  180 IF F% F%=FALSE:Z%=FALSE
  190 IF G% PROClevel:Z%=TRUE
  200 UNTIL L%=50
  210 VDU 22,7,10:*FX 229
  220 CALL &900:*FX15,1
  230 END
  240 :
 1000 DEF PROCgame
 1010 T%=TIME DIV 6000:M$=RIGHT$("00"+ST
R$T%,2):T%=(TIME MOD 6000)DIV 100:S$=RIG
HT$("00"+STR$T%,2):IF (TIME MOD 6000)DIV
10 =0 SOUND 1,-10,200,2
 1020 COLOUR 3:PRINTTAB(14,0);M$;":";S$;
 1030 FOR D=0 TO 100:NEXT:*FX15,0
 1040 H%=X%:V%=Y%
 1050 IF INKEY(-98) AND X%>1 PROCleft
 1060 IF INKEY(-67) AND X%<18 PROCright
 1070 IF INKEY(-73) AND Y%>2 PROCup
 1080 IF INKEY(-105) AND Y%<28 PROCdown
 1090 IF INKEY(-113) AND NOT INKEY(-2) F
%=TRUE:ENDPROC
 1100 IF INKEY(-113) AND INKEY(-2) THEN
210
```

Continued on page 56

Printing Fractions

Stephen Colebourne makes this a little more straightforward from Wordwise and Interword.

FRAC will print fractions from Wordwise or Interword. It is used as a star command with two parameters, the numerator and the denominator of the fraction. e.g. *FRAC 3 7 will print in superscript 3 followed by subscript 7 - three sevenths. It will accept one parameter if it is a MODE7 fraction character.

Type in the listing, save as anything but *Frac*, then run it. You will be asked if you want the code saved to disc, as long as there have been no errors in assembly type 'Y' here. The code is saved to the disc as *Frac* and, as it is run from the disc rather than from memory, you will need a copy of it on your word processor work disc.

In Wordwise you access the facility by making it an embedded (green) command where ever you want it in the text. This is done in edit mode by pressing f1, typing *FRAC 3 4 (for example), followed by f2. In Interword it will need to be part of the 'Embedded Commands' menu accessed via f1. Move the cursor to the place in the text you want your fraction, press f1, move the choice bar to the '*' line and enter FRAC 3 4 then press Escape. Embedded commands in Interword can be rather difficult to edit once they're in the text; see page 93 of the manual for details. You must make sure the *FRAC file is on the disc in the drive when you are printing out as this when it is accessed. In ADFS it would need to be in the current directory, though you could try putting it in the root and calling it with *$.FRAC.

```
 10 REM Program Frac
 20 REM Version B 1.0
 30 REM Author  Stephen Colebourne
 40 REM BEEBUG  April 1993
 50 REM Program subject to copyright
 60 :
100 oswrch=&FFEE:osbyte=&FFF4
110 osword=&FFF1:osargs=&FFDA
120 osbget=&FFD7:osfind=&FFCE
130 FOR T=0 TO 3 STEP 3
140 P%=&900
150 [
160 OPT T
170 LDA#23:JSRoswrch
180 LDA#255:JSRoswrch
190 LDA#126:JSRoswrch
200 LDA#195:JSRoswrch
210 LDA#223:JSRoswrch
220 LDA#199:JSRoswrch
230 LDA#223:JSRoswrch
240 LDA#223:JSRoswrch
250 LDA#126:JSRoswrch
260 LDA#0:JSRoswrch
270 LDA#&75:JSRosbyte
280 TXA:AND#1:BNEprinteron
290 LDA#255:JSRoswrch
300 RTS
310 .printeron
320 LDA#1                \ GET ADDRESS
330 LDX#&98
340 LDY#0
350 JSRosargs
360 LDY#255
370 JSRsearch
380 CMP#92:BEQf12           \ =\
390 CMP#123:BEQf14          \ ={
400 CMP#125:BEQf34          \ =}
410 CMP#58:BCCs:JMPbad:.s   \ >9
420 STA&9A
430 JSRsearch
440 CMP#58:BCCt:JMPbad:.t   \ >9
450 STA&9B
460 JMPprint
470 .f12
480 LDA#1:STA&9A:LDA#2:STA&9B
490 JMPprint
```

```
500 .f14
510 LDA#1:STA&9A:LDA#4:STA&9B
520 JMPprint
530 .f34
540 LDA#3:STA&9A:LDA#4:STA&9B
550 .print
560 LDA#1:JSRoswrch          \SUPERSCRIPT
570 LDA#27:JSRoswrch
580 LDA#1:JSRoswrch
590 LDA#83:JSRoswrch
600 LDA#1:JSRoswrch
610 LDA#0:JSRoswrch
620 LDA#1:JSRoswrch
630 LDA&9A:JSRoswrch
640 :
650 LDA#1:JSRoswrch          \SUBSCRIPT
660 LDA#27:JSRoswrch
670 LDA#1:JSRoswrch
680 LDA#83:JSRoswrch
690 LDA#1:JSRoswrch
700 LDA#1:JSRoswrch
710 LDA#1:JSRoswrch
720 LDA&9B:JSRoswrch
730 :
740 LDA#1:JSRoswrch          \CANCEL
750 LDA#27:JSRoswrch
760 LDA#1:JSRoswrch
```

```
 770 LDA#84:JSRoswrch
 780 :
 790 LDA#3:JSRoswrch
 800 LDA#255:JSRoswrch
 810 LDA#2:JSRoswrch
 820 RTS
 830 .search
 840 INY
 850 LDA(&98),Y
 860 CMP#13:BEQbad
 870 CMP#32:BEQsearch
 880 CMP#49:BCCbad
 890 CMP#127:BCSbad
 900 RTS
 910 .bad
 920 BRK
 930 EQUB14
 940 EQUS"Bad Fraction"
 950 BRK
 960 ]:NEXT
 970 PRINT;~P%;
 980 :
 990 PRINT"Save code? (Y/N) ";
1000 q$=GET$
1010 IF q$="Y" ORq$="y" THEN *SAVE FR
AC 900 AFF
1020 END
```

League Tables

League Secretary's suffering from sleepless nights?
Phil Gisby could have the cure.

This program evolved from an exercise in using procedures and data handling. Having developed the idea for a fairly simple example, it was quickly put to a practical use.

It collates match results and produces league tables which can be viewed on-screen or printed out. It caters for the number-crunching part of league tables, but not the 'validating' of match results before they are accepted for inclusion.

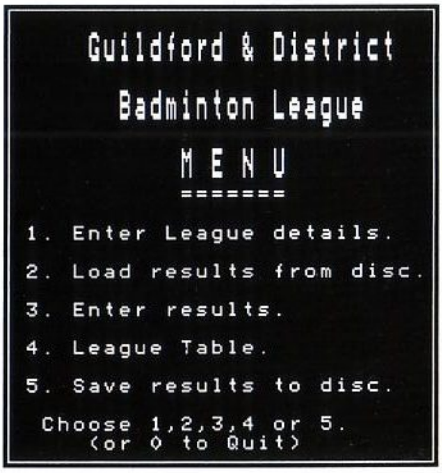
How the league is set up, and the rules governing how positions are resolved, is taken from the League Rules of whoever utilises the program. This version is for a particular Badminton League, but could easily be adapted to other applications.

The program should be typed in and saved as normal. As it uses computed GOTOs the line numbers must be exact. Lines 2050-2120 contain the decision-making process for positional changes. Line 2070 restricts the process to one division at a time. Line 2080 compares total points scored, and line 2090 compares total games won in the event of the previous figures being equal. The REPEAT:sort%=0 (line 2050).....UNTIL sort%=0 (line 2110) loop continues the process until no further changes are required.

The sleuths amongst you will enjoy working out what the other records stand for, and how they could also be used if required.

The tables are stored as files separate to the program. This means that no alterations are needed to re-use it each season, and out-of-date files can be erased.

Details for the League are entered as a one-off exercise at the start of each season. Once a set of results has been entered, positions are adjusted automatically. Errors or late adjustments can also be catered for.



```
  Guildford & District
    Badminton League
        M E N U
        =======
1. Enter League details.
2. Load results from disc.
3. Enter results.
4. League Table.
5. Save results to disc.
 Choose 1,2,3,4 or 5.
     (or 0 to Quit)
```

Main menu

The general principle which the program works on is that options from the main menu are treated as branches from which the various procedures are called. This makes it possible to keep returning to this menu with the minimum of GOTO's, or any incomplete procedures.

The following is based upon the options available from the main menu:

1. ENTER LEAGUE DETAILS

First enter a title for the league division. Team names are then entered - the limit of 15 characters is to fit in with screen displays. After a name has been typed in, press Return and you are invited to (R)epeat the same division, start a (N)ew division or return to the Menu (Space Bar).

Details should be entered in sequence, since teams are numbered as they go in. These numbers are used in the running of the program, and out-of-sequence entries may produce some strange effects!

2. LOAD RESULTS FROM DISC

To load in previously recorded results from option 5, and where you start once the leagues have been established.

The use of the date as a results file name is recommended - with obvious benefits, but anything can be used within the limits for file names. For the example shown here, in item 4 I have used *8Apr92*. A sample file, *Final*, is included on this month's disc.

3. ENTER RESULTS

A sub-menu provides options for adding and removing results, or removing a team completely. Adjustments are made by removing and re-entering a result.

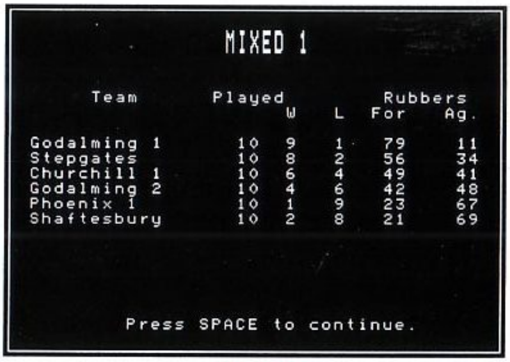
Running totals are kept of matches played, wins, losses and points (rubbers) for or against. (N.B. Badminton matches do not end in draws, but the program could cater for them.)

The program is best suited for bulk updates rather than just a few at a time. The (R)epeat and (N)ew division key options are used, as when first entering the league details.

It is during this section that the internal numbers are used - the only time these figures are seen on-screen.

4. LEAGUE TABLE

Provides the facility to view league positions on-screen, or give a complete print-out. A sample of the former is shown below.



```
                 MIXED 1

   Team         Played           Rubbers
                      W     L    For    Ag.

Godalming 1       10  9     1    79     11
Stepgates         10  8     2    56     34
Churchill 1       10  6     4    49     41
Godalming 2       10  4     6    42     48
Phoenix 1         10  1     9    23     67
Shaftesbury       10  2     8    21     69



         Press SPACE to continue.
```

A league

Processing occurs during this part of the program, so there is a short delay before the choices appear. It is worth checking through the table, after updating, before saving and/or printing to check whether any human errors have crept in.

5. SAVE RESULTS TO DISC

After updating with the latest results. Again the use of a date is recommended for file-names.

PROCEDURES

PROCon/PROCoff: turns flashing cursor on/off as required.

PROCdouble: creates Teletext double-height screen titles.

PROCheader: on-screen column headings for league divisions.

PROCprint: directs league table data to parallel printer output.

PROCpage: prints title of league division.

PROCall: produces screen display of complete league table, one division at a time.

PROCsort: reverses the position of two adjacent records, when called for.

PROChead: as PROCheader, for printer output.

PROCadd: for adding match results.

PROCremove: for removing match results.

PROCzap: to remove a team from the league - but only if no results are present.

SUMMARY

Although this version works for a Badminton League, the principles apply to virtually any league system.

The size of the table, for the number of teams and fields per record, is catered for in the array statement at line 80. In this case, there are two strings (team and division name), five numeric (games played, won and lost, points for and against) for up to 150 records. The last two - t$(2) and r(5) - are used to 'park' records during the sort routine.

If, for example, the table needed to be modified to cater for 200 records - and include statistics for drawn results - the array statement (line 80) would read:

```
DIM team$(200,2),result(200,6),t$(2),r(6)
```

It would then need a careful check through all the routines that process, display, print, load and save the data to cater for the extra field. Those venturing into this would be wise to suspend line 90 for the duration of the exercise!

A large League could be sub-divided into more manageable parts and the data saved as separate files, to be handled by the program at different times. This method is in use on an IBM-compatible PC, using the program via a BBC Basic emulator.

Using procedures makes the addition of extra features quite easy. The 'Enter Results' option, for example, only catered for adding results initially. It is only a matter of time before I come up with more routines worth including.

Error-trapping features to prevent incorrect key-strokes are incorporated, but if this fails - the Escape key returns you to the main menu with only the loss of any current on-screen entries.

Anyone with experience of collating league results by hand will appreciate the difference a system like this makes. Within the League it is in use for, the task went from a nightly ordeal to about an hours work per evening - once or twice a month.

```
   10 REM Program League Tables
   20 REM Version B 1.3
   30 REM Author  P.E.Gisby
   40 REM BEEBUG  April 1992
   50 REM Program subject to copyright
   60 :
   70 :
   80 K%=0:DIM team$(150,2),result(150,5
),t$(2),r(5)
   90 ON ERROR GOTO100
  100 MODE7:REM **  Main Menu  ********
  110 PROCon:PROCdouble(134,"Guildford &
 District",2):PROCdouble(134,"Badminton
League",5)
  120 PROCdouble(130,"M E N U",8):PRINTT
AB(16,10)CHR$130;"======="
  130 PRINTTAB(6,12)CHR$134;"1. Enter Le
```

```
ague details."
  140 PRINTTAB(6,14)CHR$134;"2. Load res
ults from disc."
  150 PRINTTAB(6,16)CHR$134;"3. Enter re
sults."
  160 PRINTTAB(6,18)CHR$134;"4. League T
able."
  170 PRINTTAB(6,20)CHR$134;"5. Save res
ults to disc."
  180 PRINTTAB(10,23)CHR$131;"(or 0 to Q
uit)"
  190 PRINTTAB(7,22)CHR$131;"Choose 1,2,
3,4 or 5. ";
  200 REPEAT A=GET:UNTIL A>47 AND A<54
  210 IF A=48 THEN 230
  220 GOTO(A-48)*500
  230 MODE7:END
  490 :
  500 CLS:REM  New Teams
  510 PROCdouble(134,"New Teams",2)
  520 PRINT''TAB(11)CHR$131;"Division";:
INPUT" - "D$
  530 PRINT'CHR$131;"Enter team name - t
hen press R(repeat),"
  540 PRINTTAB(5)CHR$131;"N(new division
) or SPACE(menu):"
  550 PRINT'TAB(7)STRING$(15,"_")
  560 VDU11:INPUTTAB(7)T$
  570 REPEAT A=GET:UNTIL A=32 OR A=78 OR
 A=82
  580 K%=K%+1:team$(K%,1)=T$:team$(K%,2)
=D$
  590 IF A=82 THEN 550 ELSE IF A=78 THEN
 500 ELSE IF A=32 THEN 100
  990 :
 1000 REM  Load Data from disc
 1010 CLS:PROCdouble(130,"Load Results",
2)
 1020 PRINTTAB(6,8)CHR$131;"Enter ";:INP
UT"last file-name: "d$
 1030 PROCoff:PRINTTAB(12,10)CHR$134;CHR
$136;"LOADING"
 1040 A=OPENUP(d$)
 1050 REPEAT:K%=K%+1
 1060 INPUT#A,team$(K%,1),team$(K%,2),re
sult(K%,1),result(K%,2),result(K%,3),res
ult(K%,4),result(K%,5)
 1070 UNTIL EOF#A
 1080 CLOSE#A
 1090 PRINT''TAB(6,10)CHR$134;"Results f
or ";d$;" now loaded."
 1100 PRINT'TAB(6)CHR$131;"Press SPACE t
o continue."
 1110 REPEATUNTIL GET=32:GOTO100
 1490 :
 1500 REM  Results
 1510 PRINT'TAB(8)CHR$131"Which";:INPUT"
 Division? "D$
 1520 CLS:PROCdouble(130,"Results",2)
 1530 PROCdouble(130,D$,5):PROCoff
 1540 PRINTTAB(10,10)CHR$134;"1. Add res
ults"
 1550 PRINTTAB(10,12)CHR$134;"2. Remove
results"
 1560 PRINTTAB(10,14)CHR$134;"3. Remove
team"
 1570 PRINTTAB(10,16)CHR$134;"4. Return
to Main Menu"
 1580 PRINTTAB(10,20)CHR$131;"Choose 1,2
,3 or 4";
 1590 REPEAT A=GET:UNTIL A>48 AND A<53
 1600 IF A=49 PROCadd ELSE IF A=50 PROCr
emove ELSE IF A=51 PROCzap ELSE IF A=52
THEN 100
 1610 PROCoff:PRINTTAB(9,22)CHR$131;"Pre
ss R,N or SPACE       "
 1620 REPEAT B=GET:UNTIL B=32 OR B=78 OR
 B=82
 1630 IF B=82 THEN 1520 ELSE IF B=78 THE
N 1510 ELSE IF B=32 THEN 100
 1990 :
 2000 REM  Tables
 2010 CLS: PROCdouble(134,"League Tables
",2)
 2020 PRINT''CHR$134;"Options :"
 2030 PRINT'CHR$134;"1. Look at Division
s."
 2040 PRINT'CHR$134"2. Print out complet
e League.":PROCoff
 2050 REPEAT:sort%=0
 2060 FOR T%=1TOK%-1
 2070 IF team$(T%,2)<>team$(T%+1,2) THEN
 2100
 2080 IF result(T%,2)<result(T%+1,2) THE
N PROCsort(T%)
 2090 IF team$(T%,2)=team$(T%+1,2) AND r
esult(T%,2)=result(T%+1,2) AND result(T%
,4)<result(T%+1,4) THEN PROCsort(T%)
 2100 NEXT
 2110 UNTIL sort%=0
 2120 PROCon
 2130 PRINTTAB(5,13)CHR$131;"Choose 1,2
or SPACE. ";
 2140 REPEAT A=GET:UNTIL A=49 OR A=50 OR
 A=32:IF A=32 THEN 100
 2150 IF A=50 PRINT''CHR$131;"Is Printer
 connected? (Y/N) " ELSE 2170
 2160 PROCoff:A=GET:IF A<>89 THEN 2000 E
LSE PROCprint:GOTO100
 2170 PROCon:PRINT''TAB(5)CHR$131;"Enter
";:INPUT" Division or ALL: "D$
 2180 IF D$ <>"ALL" THEN 2190 ELSE PROCa
ll:GOTO2000
 2190 CLS:PROCdouble(130,"League Table",
2):PROCdouble(130,D$,5):PROCoff
 2200 PRINT':PROCheader:PRINT
 2210 FOR T%=1TOK%
 2220 IF team$(T%,2)<>D$ THEN 2240
 2230 PRINTCHR$134;team$(T%,1);TAB(18);r
esult(T%,1);TAB(22);result(T%,4);TAB(26)
;result(T%,5);TAB(30);result(T%,2);TAB(3
6);result(T%,3)
 2240 NEXT
 2250 PRINTTAB(10,22)CHR$131;"Press R or
 SPACE."
 2260 REPEAT A=GET:UNTIL A=32 OR A=82
 2270 IF A=82 THEN 2170 ELSE 100
 2490 :
 2500 REM  Save Data
 2510 CLS:PROCdouble(130,"Save Results",
2)
 2520 PRINT''TAB(6)CHR$131;"Enter";:INPU
T" new file-name - "d$:PROCoff
 2530 PRINT'CHR$131;"Then press SPACE to
 continue.":REPEATUNTIL GET=32
 2550 PRINT:A=OPENOUT(d$)
 2560 FOR C%=1TOK%
 2570 PRINT#A,team$(C%,1),team$(C%,2),re
sult(C%,1),result(C%,2),result(C%,3),res
ult(C%,4),result(C%,5)
 2580 NEXT:CLOSE#A
 2590 PRINT''CHR$134;"Results for ";d$;"
 now saved."
 2600 PRINT'TAB(7)CHR$131;"Press SPACE t
o continue."
 2610 REPEATUNTIL GET=32:GOTO100
 2990 :
 3000 REM ******** Procedures ********
 3010 :
 3020 DEF PROCon:VDU23;10,&71;0;0;0;:END
PROC
 3030 :
 3040 DEF PROCoff:VDU23;10,32;0;0;0;:END
PROC
 3050 :
 3060 DEF PROCdouble(C,T$,P):X=INT((38-L
EN(T$))/2)
 3070 FOR Y=P TO P+1:PRINTTAB(X,Y)CHR$14
1;CHR$(C);T$
 3080 NEXT:ENDPROC
 3090 :
 3100 DEF PROCheader
 3110 PRINTTAB(5)CHR$130;"Team";TAB(16)"
Played";TAB(30)"Rubbers"
 3120 PRINTTAB(21)CHR$130;"W   L";TAB(29
)"For   Ag.":ENDPROC
 3130 :
 3140 DEF PROCsort(X%):sort%=sort%+1
 3150 t$(1)=team$(X%,1):t$(2)=team$(X%,2
):FOR R%=1TO5:r(R%)=result(X%,R%):NEXT R
%
 3160 team$(X%,1)=team$(X%+1,1):team$(X%
,2)=team$(X%+1,2):FOR R%=1TO5:result(X%,
R%)=result(X%+1,R%):NEXTR%
 3170 team$(X%+1,1)=t$(1):team$(X%+1,2)=
t$(2):FOR R%=1TO5:result(X%+1,R%)=r(R%):
NEXT R%
 3180 ENDPROC
 3190 :
 3200 DEF PROCprint:PROCoff
 3210 *FX5,1
 3220 *FX3,10
 3230 PRINT CHR$8"Guildford & District B
```

Continued on page 54

Public Domain Software

Alan Blundell introduces more new PD software this month.

I've been busy this month cataloguing some more new releases and thought that some of them were well worth a mention in this column.

The first is a disc of hints and tips on the BBC Micro, sent to me by Glyn Fowler but compiled by Richard Sterry from the archives of the Wakefield BBC Micro User Group. This is perhaps the most comprehensive collection of short hints and tips I have seen. Many of the hints have been fairly well known for some time and many may duplicate some of those which have appeared in BEEBUG in the past. Even so, it's handy to have them all organized together in one place, ready to be dipped into when you feel the urge to extend your expertise, and one or two of the tips were completely new to me. In all, there are over 200 of them, collected in plain ASCII text files containing a few hints each. The disc contains over 270K of text in all, including the excellent indexes.

Also by Richard Sterry (G4BLT) is a disc of radio amateur utilities. From correspondence, I think this is quite a popular interest, but one which has not been particularly well served by PD on the Beeb up to now. Whilst I am not well qualified to judge, knowing little about radio, this software has been described to me as being of 'near professional quality'. Given my lack of expertise, it's probably better for me to detail the software without comment of my own. The disc includes:

Alldump	- ROM/RAM memory dump utility
CW/qso	- CWtransceive program. Needs simple tone decoder.
Demo	- demo file for use with Mfile.
Epson	- machine code for `/# sign clash on Epson-type printers.
Locate	- comprehensive locator and contest scoring program.
MasTerm	- TNC driver program for the Master (or Compact with RS423 interface).
Mfile	- a card-index database
Moonloc	- program for predicting the position of the moon in the sky.
Morse	- comprehensive morse-tutor program with key practice facility.
RiseSet	- sunrise/sunset predictor for the Master 128.
Rtty	- RTTY transceive program, which requires an external TU.
SunLoc	- program for predicting the position of the sun in the sky.
UUBEEB	- the original disc-based version of the UU encoder/decoder software, by G7GLN.
UUROM	- the newer ROM or Sideways RAM version of UUBEEB
PJJTERM	- the TNC driver program (v3.37) for the BBC B/B+/ Master, by G1PJJ.

If all of that makes some sort of sense to you, it may be worth a look.

REMINISCENCES OF AN OLD TIMER

I have now had a bit of time to take a look again at the software from some of the early BEEBUG magazine discs, between 1984 and early 1987. There are plenty of gems, of course, given the quality of the listings published in BEEBUG, but I came across some particular old favourites.

Utilities and application programs which you find genuinely useful on a day to day basis never get forgotten about because they become part of your routine. They may include a print spooler, a Basic program compacter or a disc menu program. Such programs are either useful in the sort of computing you do or not. There are certainly plenty of utilities in the history of BEEBUG. Apart from those already mentioned there were many others between '84 and '87, including a RAM tester, a DFS 'move down' routine, screen freeze/save, Basic cross-referencer, Basic program autorun, Basic 'pretty-printer', split screen utility, DFS 40T/80T conversion, disc benchmarks, function key editor, picture compression, ROM controller, View printer drivers, Master CMOS RAM clock controller, 6502 disassembler, disabling the Break key, DFS and ADFS disc menus aplenty; and windows and icons.

Closely related to utilities are those programs which demonstrate a particular technique or function, like accurate arithmetic in Basic or assembler, interrupt-driven music, colour fill graphics (not built into the Model 'B', unlike the Master series), mixing screen modes, graphics using polar co-ordinates, multiple windows, using Wordwise Plus segments, plotting graphics in mode 7, using recursion, producing a Basic language compiler, programming sideways ROM/RAM, disc recovery or inbetweening graphics.

During the mid-80s, BEEBUG included at least one game in each issue. There isn't a lot of point in merely listing their names, but many were implementations of commercial games which were popular at the time. Considering the proportion of Basic code in most of the games, it is surprising how playable most of them are. These programs are perhaps the most striking examples of using the programming method which best suits your end goal: use Basic, which is quick and easy to write and debug, for the bulk of the program, but use assembler for the critical parts where speed of operation is essential. Have you ever thought how different the history of the BBC Micro would have been if there had been no built-in 6502 assembler? Of course there are several other ways to generate machine coded software, and commercial software probably wouldn't have been any different, but magazine listings and little programs that people knocked together and then gave to their friends would have been completely different. For one thing, programming for sideways RAM would never have caught on.

BEEBUG has always had its share of serious software applications, like temperature measurement, domestic accounts, spreadsheets, flowchart generation, a pop-up calculator, loan repayments, data storage, computer simulation and chart recording to name a few. But the programs which were most evocative of memory when I looked at them again were those which I remembered as individual programs, rather than examples of a type like 'spreadsheet' or 'game' or whatever. Most of the items I latched onto in this way are, I suppose, really just demos - if by 'demo' we mean things you just watch and/or listen to, rather than use or play to beat the high score. These programs appeal to the computer buff in us, or at least those of us whose main use for their computer is messing about, learning about the thing for its own sake, and playing the odd game. Put like that, it doesn't sound like much of a justification for owning a computer, but I believe that something like that is at least part of the reason the BBC Micro became so popular and has continued to be popular long after other machines which originated in the early 80s have become expensive bookends. The Beeb just has so many built-in bits to

Continued on page 23

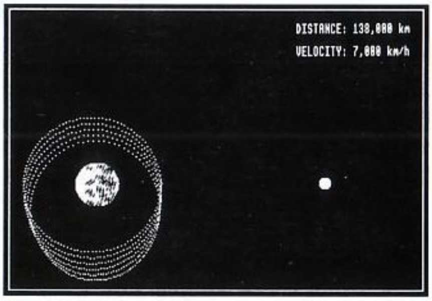
Gravity and Orbits (Part 2)

It's one small step for Cliff Blake.

This month's orbit program, listed below, takes us to the Moon and back. Type the listing in and save it as 2-lunar.

PLANET AND MOON ORBIT

As stated last time, bodies are oversize in relation to their spacing, and mass ratios are reduced. This makes planet and moon orbit simulation a little more tricky than simple Earth orbit.



```
DISTANCE: 138,000 km
VELOCITY: 7,000 km/h
```

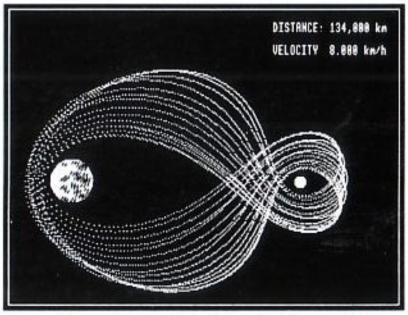
Staying close to home

THE PROGRAM

A choice is given as to whether the initial orbit of the spaceship shall be circular or figure-of-eight. The starting direction is random, so that it may be up the screen or down. Again the track is marked to show its shape, while velocity and distance from the centre of the Earth are displayed.

EFFECT OF MOON'S GRAVITY

Select the circular orbit, and just let the program run. You will notice that the orbit becomes eccentric and elliptical due to the pull accelerating the spaceship when it is moving towards the Moon, and slowing it when moving away. After ten orbits the vehicle will crash to Earth. Either before or after the crash, you can press key R to repeat the circular orbit. . .



```
DISTANCE: 134,000 km
VELOCITY 8.000 km/h
```

Sick astronauts not quite landing

FIGURE-OF-EIGHT TO CIRCULAR

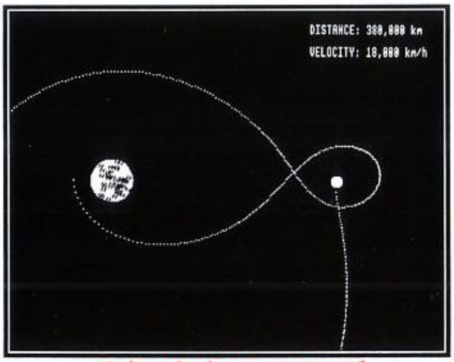
Press Escape, and run the complete program again. This time select 'Figure-of-eight'. You can go for tea while a number of orbits build up to show how they vary. Press 'R' to repeat, and allow almost one complete orbit to take place then, while the spaceship is on the side of the Earth away from the Moon, press the '<' key to slow it down. If you judge correctly, it should orbit the planet only, and a little adjustment as in the previous article, can produce a low circular orbit without crashing. This exercise is comparatively easy.

Squashed LEMming

CIRCULAR TO FIGURE-OF-EIGHT

Press Escape, and run the complete program again, selecting circular orbit.

This time press the '>' key to accelerate the spaceship when it is furthest from the Moon. The aim is to achieve a figure-of-eight orbit. This seems impossible at first, but it becomes easier if you practice all-night! A slight acceleration or retardation on the journey may be useful. You can always press 'R' to try again.



```
DISTANCE: 380,000 km
VELOCITY: 10,000 km/h
```

Going the long way round

CIRCLING THE MOON

If you can achieve a curve going round the Moon, pressing '<' key may slow the spaceship sufficiently, so that it orbits the Moon. Due to the strong influence of the Earth in this simulation, only a couple of such orbits are likely.

Keep training on these situations, and next time we'll look at them from the point of view of gravitational pits.

```
   10 REM Program LUNAR
   20 REM Version B2.0
   30 REM Author  Cliff Blake
   40 REM BEEBUG  April 1993
   50 REM Program subject to copyright
   60 :
  100 ENVELOPE1,8,1,-1,1,1,1,1,121,-10,-
5,-2,120,120
  110 MODE7:*FX11
  120 PROCinfo:g%=GET
  130 MODE0:VDU5
  140 REPEAT
  150 quit%=FALSE
  160 CLS:PROCplanet:PROCmoon:PROCspaces
hip
  170 REPEAT
  180 rerun%=FALSE
  190 PROCmove:PROCdistance
  200 PROCgravity:PROCthrust
  210 PROCflags
  220 UNTIL rerun%
  230 UNTIL quit%
  240 VDU4:*FX12
  250 CLS:*FX21
  260 END
  270 :
 1000 DEF PROCmove
 1010 MOVE Xs,Ys:PRINT CHR$64
 1020 ENDPROC
 1030 :
 1040 DEF PROCdistance
 1050 Xdp=320-Xs:Ydp=512-Ys
 1060 Xdm=960-Xs:Ydm=512-Ys
 1070 Rdps=Xdp*Xdp+Ydp*Ydp:Rdp=SQR(Rdps)
 1080 Rdms=Xdm*Xdm+Ydm*Ydm:Rdm=SQR(Rdms)
 1090 dis%=INT(.6*Rdp)
 1100 IF Rdp<60 OR Rdm<15 SOUND0,1,0,5:P
ROCcrash
 1110 VDU4:PRINT TAB(55,2)"DISTANCE: ";d
is%;",000 km";SPC3:VDU5
 1120 ENDPROC
 1130 :
 1140 DEF PROCgravity
 1150 Xgp=20000*Xdp/Rdp/Rdps:Ygp=20000*Y
dp/Rdp/Rdps
 1160 Xgm=2000*Xdm/Rdm/Rdms:Ygm=2000*Ydm
/Rdm/Rdms
 1170 Xv=Xv+Xgp+Xgm:Yv=Yv+Ygp+Ygm
 1180 Rvs=Xv*Xv+Yv*Yv:Rv=SQR(Rvs)
 1190 vel%=INT(.85*Rv)
 1200 VDU4:PRINT TAB(55,4)"VELOCITY: ";v
el%;",000 km/h";SPC3:VDU5
 1210 ENDPROC
 1220 :
 1230 DEF PROCthrust
 1240 Xt=0.15*Xv/Rv:Yt=0.15*Yv/Rv
 1250 IF INKEY(-103) THEN Xv=Xv-Xt:Yv=Yv
-Yt:SOUND0,-7,6,4
 1260 IF INKEY(-104) THEN Xv=Xv+Xt:Yv=Yv
+Yt:SOUND0,-7,6,4
 1270 Xs=Xs+Xv:Ys=Ys+Yv
 1280 ENDPROC
 1290 :
 1300 DEF PROCflags
 1310 IF INKEY(-52) THEN rerun%=TRUE
 1320 IF INKEY(-17) THEN rerun%=TRUE:qui
t%=TRUE
 1330 ENDPROC
 1340 :
 1350 DEF PROCplanet
 1360 Ca=COS(PI/40):Sa=SIN(PI/40)
 1370 CA=1:SA=0:MOVE 320+60,512
 1380 FOR A=1 TO 80
 1390 Cp=CA:Sp=SA
 1400 CA=Cp*Ca-Sp*Sa:SA=Sp*Ca+Cp*Sa
 1410 x=60*CA+320:y=60*SA+512
 1420 MOVE 320,512:PLOT 85,x,y
 1430 NEXT A
 1440 GCOL0,0
 1450 FOR n%=1 TO 50
 1460 MOVE 280+RND(80),452+RND(120):PRIN
T"~"
 1470 NEXT n%
 1480 GCOL0,1
 1490 ENDPROC
 1500 :
 1510 DEF PROCmoon
 1520 CA=1:SA=0:MOVE 960+15,512
 1530 FOR A=1 TO 80
 1540 Cp=CA:Sp=SA
 1550 CA=Cp*Ca-Sp*Sa:SA=Sp*Ca+Cp*Sa
 1560 x=15*CA+960:y=15*SA+512
 1570 MOVE 960,512:PLOT 85,x,y
 1580 NEXT A
 1590 ENDPROC
 1600 :
 1610 DEF PROCspaceship
 1620 VDU23,64,192,0,0,0,0,0,0,0
 1630 sign=(-1)^RND(8)
 1640 IF o%=1 Xs=500:Ys=512:Xv=0:Yv=sign
*11
 1650 IF o%=2 Xs=210:Ys=512:Xv=1:Yv=sign
*17.1
 1660 ENDPROC
 1670 :
 1680 DEF PROCinfo
 1690 y$=CHR$131:c$=CHR$134:w$=CHR$135
 1700 PRINT TAB(10,3)y$+"LUNAR ORBITER"
 1710 VDU28,0,24,39,6
 1720 PROCselect
 1730 PRINT'w$+"Gently holding down the
< key will"
 1740 PRINT w$+"fire the retro unit to s
low the ship."
 1750 PRINT'w$+"Gently holding down the
> key will"
 1760 PRINT w$+"fire the drive to accele
rate the ship."
 1770 PRINT'w$+"Press any key to start."
 1780 PRINT'w$+"Press R to clear the scr
een & Repeat."
 1790 PRINT'w$+"Press Q to Quit."
 1800 ENDPROC
 1810 :
 1820 DEF PROCselect
 1830 PRINT TAB(6)w$+"Select initial orb
it"
 1840 PRINT''TAB(3)w$+"1  Circular aroun
d planet"
 1850 PRINT'TAB(3)w$+"2  Figure-of-eight
 around"
 1860 PRINT TAB(6)w$+"planet and moon."
 1870 REPEAT:o%=GET-48:UNTIL o%=1 OR o%=
2
 1880 CLS:IF o%=1 PROCco ELSE PROCfoeo
 1890 ENDPROC
 1900 :
 1910 DEF PROCco
 1920 PRINT c$+"A spaceship is in circul
ar orbit"
 1930 PRINT c$+"around a planet. Try tak
ing it"
 1940 PRINT c$+"on a journey round the m
oon,"
 1950 PRINT c$+"and back into planetary
orbit."
 1960 ENDPROC
 1970 :
 1980 DEF PROCfoeo
 1990 PRINT c$+"A spaceship is in figure
-of-eight"
 2000 PRINT c$+"orbit around a planet an
d moon."
 2010 PRINT c$+"Try reducing the orbit t
o a circular"
 2020 PRINT c$+"one around the planet on
ly."
 2030 ENDPROC
 2040 :
 2050 DEF PROCcrash
 2060 REPEAT:UNTIL INKEY(-52):rerun%=TRU
E
 2070 ENDPROC
```

Sorting (Part 1)

by David Fell

From time to time most programmers need to sort a list of items into order. Maybe it's a set of scores, or perhaps a list of names to be put into alphabetical sequence. This month I'll give details of a couple of straightforward methods and suggest code which you could use in your own programs.

First, we'll look at the well-known and aptly-named *Bubble Sort*. Suppose we must put a listinto ascending order. The bubble sort starts with the first 2 elements, compares them and, if needed, swaps them so that the larger is in position 2. It then compares elements 2 and 3 and again puts the larger value into the higher position.

The sort continues until it reaches the end of the list when, all being well, the largest element will have reached the top. It has *bubbled* up through the list. The sort then goes back to the start and bubbles the next-largest element up to the second from top position. So it goes on until the whole list is sorted.

If there is much swapping to do, the bubble sort can be painfully slow. However, as soon as a pass through the list is made without swapping anything, the whole lot is then sorted. This means that, with only a few items out of place, the sort can be very fast indeed. The whole process is illustrated in figure 1.

H C F J A D G I E B	Initial order
C H F J A D G I E B	after 1st comparison
C F H J A D G I E B	after 2nd comparison
C F H J A D G I E B	after 3rd comparison
C F H A D G I E B J	after 1st pass
C F A D G H E B I J	after 2nd pass

Figure 1. Initial stages in a bubble sort

BUBBLE SORT

```
 10000 DEF PROCbubble(ST%,FIN%)
 10010 IF ST%>=FIN% THEN ENDPROC
 10020 LOCAL F%,I%
 10030 REPEAT
 10040   F%=FALSE
 10050   FOR I%=ST% TO FIN%-1
 10060     IF array(I%)>array(I%+1) THEN
           PROCswap
 10070     NEXT
 10080   FIN%=FIN%-1
 10090   UNTIL NOT F%
 10100 ENDPROC
 10490:
 10500 DEF PROCswap
 10510 LOCAL temp
 10520 temp=array(I%)
 10530 array(I%)=array(I%+1)
 10540 array(I%+1)=temp
 10550 F%=TRUE
 10560 ENDPROC
```

The procedure PROCbubble assumes that the data to be sorted is in *array()*. Obviously, you should use your own variable name here. The routine expects two input parameters: ST%, which defines the first element of the array to be sorted, and FIN% which defines the last. This means that you don't have to sort an entire array every time. For instance, if *array()* had 300 elements, *PROCbubble(100,200)* would sort the middle third only. The procedure makes sure the limits are sensible. The subsidiary procedure PROCswap swaps two elements when needed.

After each pass through the array, we know that the next highest value has reached its final position; FIN% is thus reduced by 1 so that we don't waste time checking the sorted items at the top of the array. F% shows if there are any swaps in a pass through the list, and allows an early exit.

The bubble sort is simple but can be slow. However, there is a much faster version known as the *Shell Sort* after its originator. This time, instead of always comparing adjacent elements, the sort starts by comparing, and swapping, items which are separated by some distance. Whenever no swaps occur in a pass, this distance is halved and the sorting starts again.

The process continues until the gap is 1, when it is just like a bubble sort. However, by the time it gets there, the list has already been sorted into rough order and the whole thing finishes very quickly. Here is a procedure to do this job.

SHELL SORT

```
11000 DEF PROCshell(ST%,FIN%)
11010 IF ST%>=FIN% THEN ENDPROC
11020 LOCAL F%,I%,S%,T%
11030 S%=2^INT(LOG(FIN%-ST%)/LOG(2))
11040 REPEAT
11050   T%=FIN%-S%
11060   REPEAT
11070     F%=FALSE
11080     FOR I%=ST% TO T%
11090       IF array(I%)>array(I%+S%)
          THEN PROCswaps
11100       NEXT
11110     T%=T%-1
11120     UNTIL NOT F%
11130   S%=S% DIV 2
11140   UNTIL S%=0
11150 ENDPROC
11490:
11500 DEF PROCswaps
11510 LOCAL temp
11520 temp=array(I%)
11530 array(I%)=array(I%+S%)
11540 array(I%+S%)=temp
11550 F%=TRUE
11560 ENDPROC
```

You can see its links with the bubble sort. At line 11030, S% is set to the initial gap value. This must be a power of 2 (so it can be continually halved as the sort progresses) and the line calculates the largest number that will fit between ST% and FIN%. T% holds the upper limit of the FOR-NEXT loop; its start value is set so that the program does not go outside the array and, as before, it is decremented on every pass.

It's hard to say how much better the Shell sort is than the bubble, since so much depends on the starting data. In general terms, though, the bigger the array, the relatively faster it is: 200 random elements are sorted about 4 times quicker, while 500 gives an advantage of around 7. It is NOT always quicker though. If you are certain that only a few - say no more than 2% - of the items in a list are misplaced, then use the bubble sort. It will probably correct them all in a single pass, whereas the Shell sort must always have at least one pass at each gap setting.

Finally, let's have a look at these two sorts in action with this code which uses memory locations and indirection operators for speed, rather than named arrays:

```
 100 MODE7
 110 P%=HIMEM+159
 120 PROCfill
 130 PRINT TAB(0,1)"Bubble Sort:"
 140 TIME=0
 150 PROCbubble(1,200)
 160 TBUB=TIME
 170 CLS
 180 PROCfill
 190 PRINT TAB(0,1)"Shell Sort: "
 200 TIME=0
 210 PROCshell(1,200)
 220 TSHL=TIME
 230 PRINT TAB(5,18) "Bubble sort: ";
     TBUB/100;" secs"
 240 PRINT TAB(5,20) "Shell sort: ";
     TSHL/100;" secs"
 250 END
 260:
1000 DEF PROCfill
1010 FOR I%=1 TO 200
1020   P%?I%=64+RND(26)
1030   NEXT
1040 ENDPROC
```

Add the two sort routines, changing every occurrence of *array(I%)* or *array(I%+S%)* to *P%?I%* or *P%?(I%+S%)* respectively. Other references to arrays should similarly be changed.

Run the program and two random 200-element byte arrays are created and sorted. However, since *P%* points to the mode 7 screen memory, the data is displayed on the screen as characters which you can see being put into order. If you increase to, say, 500 bytes rather than 200, you will see just how much the sorts slow down.

You can, of course, play all sorts of variations on this theme. Try watching the effect of Bubble and Shell sorts of arrays with only one element misplaced. Try it with everything starting in reverse order. How would you change the sorts to give the result in descending order? A complete program demonstrating the use of both sorting techniques is included on this month's magazine disc.

Next month, I'll talk about other aspects of sorting, such as a better way of sorting strings, and how to handle arrays which are bigger than the available memory.

Note: this Workshop is based on one first published in BEEBUG Vol.3 No.10.

Public Domain Software (continued from page 17)

mess around with. The other reason that some of these programs are memorable is simply that very few people would actually take the time and trouble to write them! The programs I am thinking of include, for example 'English Country Garden', which is an engaging rendition of the well known tune, complete with a garden display, butterflies and all. Or how about 'BEEBUG plays Bach'? Here we have a computer version of a well known tune, of which there are many for the BBC, but this one has grand piano, complete with pianist who smiles at his audience! These and others like 'Stonehenge' and 'the Earth from Space' were probably written as much for the programming exercise as for the end result.

That rounds off my overview of what happened in BEEBUG volumes 3, 4 and 5. A surprisingly large proportion of the software has stood the test of time and the range is wide enough that some of it will be of interest to everyone. If you are a long-time BEEBUG reader, take a look at your back issues and remember what excited you about your computer in those heady days. If your interest is more recent, just think what you missed!

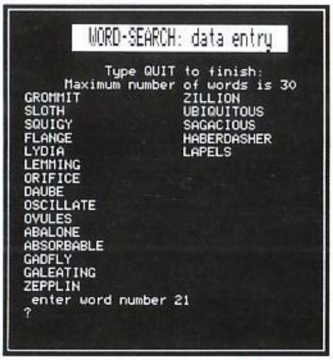
Next issue, I will try to recover from nostalgia for long enough to tell you about a few recent pieces of software that have come my way.

Note: BEEBUG programs earlier than volume 6 are only available as PD software through BBC PD, not from BEEBUG.

Word Square Upgrade

Graham Leng makes even more of your word squares.

If you enjoyed the word-search program in BEEBUG vol.11 no.7 you may like to add some extra features by adding the lines listed below to the original program. The line numbers given ensure that the lines slot in at the appropriate places in the original program. You will notice that three lines are deleted (330, 1180, 2600) and some lines are replaced (2180, 2190, 2200, 2231, 2910, 3130). Additional features are described below. This month's disc has the complete modified version on it as *Wordsq*.

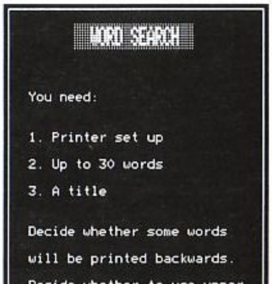


```
WORD-SEARCH: data entry

              Type QUIT to finish:
       Maximum number of words is 30
GROMMIT                  ZILLION
SLOTH                    UBIQUITOUS
SQUIGY                   SAGACIOUS
FLANGE                   HABERDASHER
LYDIA                    LAPELS
LEMMING
ORIFICE
DAUBE
OSCILLATE
OVULES
ABALONE
ABSORBABLE
GADFLY
GALEATING
ZEPPLIN
 enter word number 21
?
```

Entering the word list

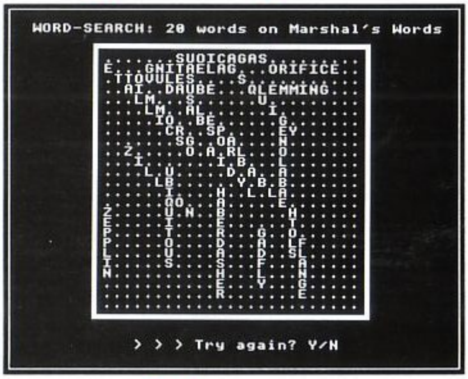
LOAD AND SAVE

You are able to save word lists after typing them in. The files are stored in directory W. Data can be loaded back in as required and used to produce new word squares. Note that completed word squares are not saved. The program uses *SPOOL and loads in using *EXEC



```
WORD SEARCH

You need:

1. Printer set up

2. Up to 30 words

3. A title

Decide whether some words

will be printed backwards.

Decide whether to use upper

or lower case

    Press RETURN to go on
```

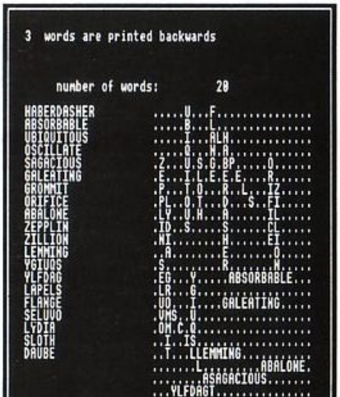
The opening screen



```
WORD-SEARCH: 20 words on Marshal's Words

.......SUOICAGAS.........
E...GNITAELAG...ORIFICE..
.TTOVULES....S...........
..AI..DAUBE...QLEMMING...
...LM...S.......U........
....LM..AL.......I.......
.....IO..BE.......G......
......CR..SP......EY.....
.......SG..OA.....N......
..Z.....O.A.RL....O......
...I.......I.B....L......
....L.U.....D.A...A......
.....LB......Y.B..B......
......I....H...L.LA......
......QO...A......E......
Z.....U.N..B.......H.....
E.....I....E.......T.....
P.....T....R...G...O.....
P.....O....D...A..LF.....
L.....U....A...D..SL.....
I.....S....S...F...A.....
N..........H...L...N.....
...........E...Y...G.....
...........R........E....
.........................

> > > Try again? Y/N
```

The key on screen

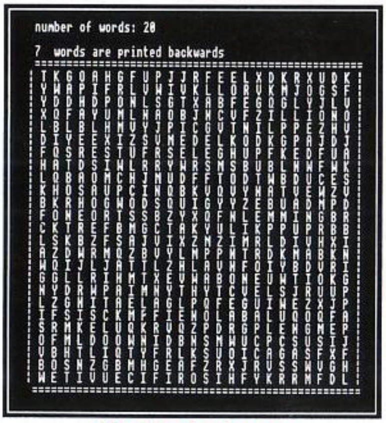
which simulates data entry from the keyboard. This uses less code and is easier to use than accessing the files via OPENIN/OUT and reading to/from the array *word$(W%)*.



```
3  words are printed backwards


    number of words:        20

HABERDASHER      .....U...F...............
ABSORBABLE       .....B...L...............
UBIQUITOUS       .....I...ALH.............
OSCILLATE        .....Q...N.A.............
SAGACIOUS        .Z...U.S.G.BP.....O......
GALEATING        .E...I.L.E.E.E....R......
GROMMIT          .P...T.O...R..L...IZ.....
ORIFICE          .PL..O.T...D...S..FI.....
ABALONE          .LY..U.H...A......IL.....
ZEPPLIN          .ID..S.....S......CL.....
ZILLION          .NI........H......EI.....
LEMMING          ..A........E.......O.....
YGIUQS           .S.........R.......N.....
YLFDAG           .EG...Y.....ABSORBABLE...
LAPELS           .LR...G..................
FLANGE           .UO...I....GALEATING.....
SELUVO           .VMS..U..................
LYDIA            .OM.C.Q..................
SLOTH            ..I..IS..................
DAUBE            ..T...LLEMMING...........
                 .......L.........ABALONE.
                 ........ASAGACIOUS.......
                 ...YLFDAGT...............
                 ..........E..DAUBE.......
                 .........................
```

The skeleton printout

One interesting effect of loading in a word list is that the program loses track of how many words are written backwards. To simplify things I have taken out the lines which print "x words are printed backwards". It would be possible to flag which words are backwards and count them when loading in but I thought this was unnecessarily complicated.



```
number of words: 20

7  words are printed backwards
==================================================
: T K G O A H G F U P J J R F E E L X D K R X U D K :
: Y W A P I F R L V W I V K L L O R V K M J O G S F :
: Y D D H D P O N L S G T X A B F E G Q G L Y J L V :
: X Q F A Y U M L W A O B J W C V F Z I L L I O N O :
: L B L B L H M V Y J P I C G V T N I L P P E Z H V :
: D I Y E E X I Z S V M E D E L K O D K G P A J D J :
: F Q S R E S T U F R S E L E G H U P F K E D F U A :
: H A T D S I W L R R Y W R S M S B U B L W W F W K :
: L Q B A O M C H J M U D F F T O D T W B D U C E S :
: K H O S A U P C I N Q B K V Q V Y W A T V E W Z V :
: B K R H O G W O D S Q U I G Y Y Z E B U A D M P D :
: F O N E O R T S S B Z Y X Q F N L E M M I N G B R :
: C K T R E F B M G C T A K Y U L I K P P U P R B B :
: L S K B Z F S A J V I X Z M Z I M R L D I V H X I :
: A Z D W R M Q Z B V Y L M P P W T R D K M A B K N :
: W Q I J L J A T T L Z H L A V W F O I Y B D V R I :
: G G L L R I H M T X E V W A B O N E U W T I O K G :
: N Y D R W P A I M W Y T H Y T C V E L S J R U U P :
: L Z G W I T A E L A G L P Q F E G U I W E Z X J P :
: I F S I S C K M F F I E W O L A B A L U Q O Q F A :
: S R M K E L U Q K R V Q Z P D R G P L E H G M E P :
: O F M L D O O W N I D B W S M W U C P C S U S I J :
: V B H T L I Q T Y F R L K S U O I C A G A S F X F :
: B O S N Z G B M H G E A F Z R X J R V S S W V G H :
: W E T I V U E C I F I R O S I H F Y K R R M F D L :
==================================================
```

The final printout

DIAGONALS

In the original program diagonals were only used from top-left to bottom-right. Now diagonals in both directions are used which produces more complicated squares.

WORD LIST

In the skeleton printout a list of words is printed on the left hand side. If the number of words exceeds 25 some words are not printed. This is corrected by the procedure *PROCextra*.

ERRORS

Disabling the Escape key with *FX229,1 makes life difficult if you want to fiddle with the program (which was probably the intention). ONERROR is used to restart the program if Escape is pressed. To escape from the program press Shift/Escape.

OTHER POSSIBILITIES

I am sure ingenious readers will be able to add other features such as loading and saving completed squares, un-reversing words, using different sized grids etc. Happy fiddling!

```
  101 ONERRORGOTO370
  242 PRINT''"Do you want to load data? Y
/N":PROCyesno:IFyes THEN PROCload
  251 PRINT''"Do you want to save data? Y
/N":PROCyesno:IFyes THEN PROCsave
  252 CLS
  270 MODE4:VDU23;8202;0;0;0;
  330
  370 VDU3:*FX15
  380 IFERR=17ANDNOTINKEY(-1)RUN
  390 MODE7:REPORT:PRINT" at line ";ERL
  400 END
  410 :
 1102 VDU23;8202;0;0;0;
 1180
 2180 IFX<25THENvert=TRUE
 2190 IFX>75THENhoriz=TRUE
 2200 IFX>25ANDX<50 THENvert=TRUE:horiz=
TRUE
 2231 IFNOThoriz ANDNOTvert THEN rowinc%
=1:colinc%=-1:row%=RND(size%-wordlen%):c
ol%=RND(size%-wordlen%)+wordlen%
 2600
 2910 IFnum%>size% PROCextra
 2931 DEFPROCextra
 2932 FORR%=size%+1TOnum%
 2933 PRINTTAB(5)word$(R%)
 2934 NEXT:ENDPROC
 3130 PRINTSPC(8)"number of words: ";num
%''
 3520 :
 3530 DEFPROCsave
 3540 F$=FNfilename
 3550 OSCLI("SPOOL W."+F$)
 3560 PRINTtitle$;CHR$13;"Y";
 3570 FORI%=1TOnum%
 3580 PRINTword$(I%);CHR$13;"Y";
 3590 NEXT
 3600 *SPOOL
 3610 CLS
 3620 ENDPROC
 3630 :
 3640 DEFPROCload
 3650 F$=FNfilename
 3660 OSCLI("EXEC W."+F$)
 3670 CLS
 3680 ENDPROC
 3690 :
 3700 DEFFNfilename
 3710 CLS
 3720 REPEAT:INPUT"Filename? "F$
 3730 UNTILLEN(F$)<8
 3740 =F$
```

Troubleshooting Guide (Part 4)

Gareth Leyshon sorts out your discs and explains the Disc Filing System.

The standard BBC disc system, present on all Masters (except the Compact) and most Beebs, is called the Acorn DFS (Disc Filing System). This is a *single density* system which can use 40 or 80 tracks, allowing one side of the disc to carry 100k or 200k of data respectively.

A similar amount of data can be held on the disc's reverse surface, provided that the disc is of *double sided* quality and your drive has a lower head to read this surface. There are some discs which are reversible (Flippies), in which case they can be removed and flipped over. Such a reversible disc can be distinguished because it has write-protect notches on two opposite edges.

The Master also carries an alternative disc system, the Advanced DFS (ADFS) which I shall deal with in the next article. The Compact supports ADFS only. There are some DFS systems for the Beeb made by suppliers other than Acorn which can increase the capacity of discs.

All Masters are equipped to use disc drives. Beebs could be bought with or without the necessary chips to use discs - to find out if a machine is suitable, type *DISC. If the computer complains 'Bad Command' then it isn't suitable, but a dealer can perform the upgrade. Otherwise, you have the necessary interface fitted. The disc drive connects to the computer via the socket labelled 'Disc Drive' on the bottom of the machine - if this socket is missing then you don't have the required hardware, though it is rare to find the socket missing in all by the earliest Beebs.

Assuming you have the necessary hardware and software, you can connect the disc drive to the computer. All drives have a broad, thin *ribbon* cable which carries the data. Some take their power from the computer via a separate lead with three or four wires and a chunky plug; others are connected directly into the mains. Either way, the golden rule is to switch off the computer before inserting or removing any plug. Inspect the chunky plug on the power lead (if present) and look carefully at the large disc socket on the computer, as both connectors may suffer from bent pins. These are easily straightened with thin-nosed pliers, they are also easily broken! Looking for bent pins is a strategy to use whenever some plug-in piece of equipment fails to work.

As I mentioned with printers last time, the drive's ribbon cable must be pushed firmly into the socket, the side pegs will clip themselves around the plug when it is securely pushed home. I have encountered many cases of disc drives, printers and other plug-in devices failing to work simply because the plug was not pushed in all the way.

WE'VE GOT THE POWER

If the drive takes its power from the computer, the chunky power plug connects to the external power supply socket. It is possible for one of the pins inside the plug to become detached and slip back inside the chunk; to cure this, push the wire connected to it to return the pin to its position, then wedge something (perhaps a matchstick) into the hole from the back to secure it in position. An old drive may reach the stage when the computer resets itself whenever a drive is used; in this case get a separate power supply for the drive, to avoid overload. In this respect the Master's power supply is more robust than that of the Beeb.

When you use a drive, either by *booting* (pressing Shift-Break), from a program or issuing a disc command, it may work successfully. Sometimes, though, you will get a message such as 'Disc Fault 18 at :0 00/00' (some of the numbers may vary from those shown). There are several possible causes of such a cryptic error: if you try several discs and none work properly, the fault is probably with the drive; it may be broken, or not compatible with the discs you are using. If, however, it's a particular disc that causes the problem, there may be a flaw on that particular disc. In either case press Ctrl-Break, insert the problem disc, type *CAT and see what happens. It may be one of the following:

1. Nothing seems to happen for several seconds, then a similar error message appears. In this case, the disc probably hasn't been designed for use with the DFS (it may be ADFS - see the next article - or designed for some other computer, or possibly a brand-new disc that hasn't been *formatted* yet).

2. An error message appears straight away, or within a couple of seconds: This suggests a serious fault on the disc, especially if the error code is something other than 18 (code 0E is common); it may have to be scrapped. You cannot recover the data on it since the *catalogue* is corrupt, but you could try formatting and verifying the disc (see below) and see if it then becomes suitable for use as a blank.

3. A list of the files present on the disc appears. You could be trying to read an 80 track disc on a 40 track drive (or vice versa) - in this case, the drive will often make a ticking noise before the computer gives an error message. If the drive has a switch marked 40/80, flick the switch (check the back of the drive if there's no switch on the front). If this fails to solve the problem, it's probably a physical fault on the part of the disc containing the file you want. You can try to *verify* the disc (see below) - there may be some parts of the disc not affected by the flaw, so it may be possible to salvage some of the data from it.

DOUBLE TROUBLE

If you want to use two drives, you can either buy a double drive unit or add a second drive by using a *splitter* lead (sometimes known as a DUCK or Dualling-Up Connector Kit). If both drives take power from the computer, you'll need a power splitter lead or an extra power supply unit. In a dual setup, the individual drives are known as 0 and 1. Zero is the default drive, and you can only boot a disc if it is in drive zero. To select drive n as the current drive, type *DRIVE *n*; this is effective until the next time Break is pressed. *CAT *n* will catalogue the disc in drive n without selecting that drive as the current drive. If your disc drives are double-sided, the other side of each disc is treated as a separate drive: drive 2 is the reverse of drive 0, with drive 3 accessing the other side of the disc in drive 1.

ERRORS WE KNOW AND LOVE

Not all the error messages are of the cryptic form. Here are some of the more friendly complaints and cures:

Can't Extend: a message in a data handling program indicating that the computer is trying to add data to a particular file which is immediately followed on the disc by other data; there is thus no room to extend the file in question. The best solution is to copy the full file to a new disc.

Cat full: the disc can only hold details of 31 files in its catalogue, regardless of how short they are individually. You must delete some unwanted files or use another disc.

Disc changed: you removed the old disc before the computer had tidied up the files it was using. Put the old disc back in the drive, type CLOSE#0, replace the new disc and try to carry on from where the error message appeared. Master users see also the note on the CLOSE#0 bug, below.

Disc full: there is not enough capacity left to store your program. There may not be enough space on the disc to store your data - you'll need to use an empty disc, or at least one carrying less data. Or it might be that the disc has enough space, but it's not gathered up in a single block. *COMPACT will pool all the free space into a block - but using this command will destroy the current memory contents, so cannot be used in order to fit the current contents of the memory on to the disc. Note that when data-handling programs open a new data file, even if it is short, it will have a minimum of 16 kilobytes (16,384 bytes) of disc-space reserved for it initially. Such a new data file cannot be created if there is less than 16k is free on the disc.

Disc read only: there's a sticky label over the write-protect notch. Either you're trying to change something on a disc which mustn't be altered, or a label has been put on the write-protect notch by mistake.

Locked: a file is protected from erasure/update. See *ACCESS below.

Not enabled: you must type *ENABLE immediately before performing a backup or destroy files instruction.

Not found: You've used Shift-Break on a disc which was not designed to be booted, or some other file which a program expects to find is missing.

Open: Typing CLOSE#0 should cure the problem. See also Master note, below.

I will not take space here to discuss the syntax of disc commands which can be found in many reference books, but you should note *ACCESS *name* will "unlock" a file, that *COMPACT *m* will tidy up the free space on the disc in drive *m* (destroying the current memory contents in the process) and *COPY *s d name* copies the *named* file from source drive *s* to destination drive *d*.

FORMATTING

To prepare a blank disc for use with the DFS, it must be formatted. Newer versions of the DFS have a built in formatter. Type *HELP DFS - if the word FORM appears somewhere in the list, you've got one built into your system. Type *FORM 40 n or *FORM 80 *n* to format to 40 or 80 tracks the disc in drive *n*, where *n* is 0, 1, 2, or 3. If you have a 40/80 switchable drive, make sure the switch is in the right position before you start. If FORM doesn't appear in your list (all Masters have it, some Beebs don't) you will need the format program on the utility disc supplied with the drive. Using *FORM or loading the format program will destroy whatever happens to be in memory at the time.

A companion command, *VERIFY *n* (or its equivalent on the utility disc) will check the disc in drive *n* to see if it has been formatted correctly. The verifier counts through the tracks on the disc - it may fail outright at an error or else display question marks against difficult-to-read tracks.

If you format a disc to a lower specification than it is capable of carrying, you lose nothing except value for money (e.g. a disc which says 96tpi - 96 tracks per inch, suitable for 80 track use - on its label can be formatted to 40 tracks, but that's not such good value as formatting to 80 tracks). If you format a disc to tougher standards than the label says, the computer might not complain initially, but you are running a big risk with your data, which could corrupt later.

BE KIND TO YOUR DISCS

Generally, discs must be treated with care. They should not be exposed to dust, heat, cold or magnetic fields. It is very easy, especially in a school situation, for discs to be left on top of the disc drive, the monitor, the computer, the cassette recorder... all of which generate magnetic fields. On no account switch the computer or disc drive on with a disc sitting in the drive. Little fingers poking about through the disc window are a sure cause of trouble: always keep discs in their sleeve when not in the drive. Keeping a good store of backups is essential! Put your master copy in storage, if possible in a metal cabinet, also store a backup of the master, and use a working copy for day-to-day purposes.

To make a backup, use the command *BACKUP 0 1 (which copies from the disc in drive 0 to the disc - blank or with unwanted contents, but it must already be formatted - in drive 1) or else use *BACKUP 0 0 if you have only one drive, when you will be prompted to change discs. It is wise to test the backup as soon as it is made, to avoid putting a faulty copy in storage. To avoid accidentally copying the blank disc on to the original, temporarily cover the latter's write-protect notch.

WHICH CHIP?

All Beebs and Masters with a disc interface carry a chip containing the disc Filing System (DFS). The original version used a chip called an 8271 to control the drive, and came with a floppy disc of utility programs. This has been replaced in modern Beebs (and *all* Master Series computers) with a 1770 type chip, and some utilities which were on the disc are now built into the computer.

With a 1770 system, if you need to access a 40 track disc on an 80-only drive, say on drive 1, then the *DRIVE command is extended to *DRIVE 1 40. To go back to drive 0 in ordinary 80 mode, use *DRIVE 0 80. There is, however, no way on any system of reading 80 track discs using a 40-track only drive.

N.B. Some Masters include a slightly better chip called the 1772 (though the on-screen message still says 1770) which can make the disc drive run faster. Some older drives may not be able to run fast enough for the 1772; when you try to load a program the disc drive will click repeatedly and you will get an error code 18 (or code 50 on ADFS). There are two solutions: have your dealer replace the 1772 with a 1770 - or get a newer disc drive. You can marginally affect the speed at which a disc drive runs with the FDRIVE configuration. Use:

```
*CONFIGURE FDRIVE n.
```

Putting n=0 or n=1 will work for most drives. Putting n=2 or n=3 will give a slower speed if you have the 1770 chip, and this is essential for older drives. These values will, however, give a faster speed with a 1772. Try the different settings with your drive to find the optimum, and if an old drive refuses to work, check whether you have a 1772 when you need a 1770.

There's another configuration setting you might note: If you put *CONFIGURE BOOT the computer will reverse the actions of Break and Shift-Break, with the result that Break or switching on will attempt to boot a disc, while Shift-Break will not. *CONFIGURE NOBOOT is the reversing command. The main use is to force the computer to boot up as soon as switched on; powering-up with a disc in the drive is not recommended, and this facility is mainly used if you have a hard disc.

THE CLOSE#0 BUG

There is a known fault in the Master's DFS system that affects some database

Continued on page 55

The Sideways Poet (Part 2)

David Houlton completes his poetry generator and he will never, never surrender.

Last month I described how a sick sense of humour, inflamed by alcohol, spawned that depraved and unnecessary ROM image *The Sideways Poet*. This month I am making good my threat to entice you deeper into the mire by explaining how you can install a new 'poet' of your own.

It should easily be possible for readers with no knowledge of machine code to install a new 'poet'. I hope that if you look closely at the way the existing data is presented, you'll find you don't need to understand the 'machinery', you just copy in a few lines here and there and add your data in a simple format. Mind you, you might gain some insight into the workings of the assembler as you go along.

Before you start, you might make life easier for yourself by setting up some of the function keys to do various jobs for you. For example:

```
   *KEY 0 SAVE "POET_C"|M
```

POET_C being merely one suggestion for a filename - it must not be the same as the filename of the original version. Use key f0 to save your work at frequent intervals. Also...

```
   *KEY 1 EQUS "
   *KEY 2 OLD|M MODE 7|M LIST|N|M
   *KEY 3 RUN|M
```

YOUR FINEST HOUR

Let's suppose that the 'poet' you want to add is actually a piece of prose, Churchill's speech "We shall fight on the beaches...." The first thing would be to add Churchill to the menu in the gap between Burns and Shakespeare. Just copy the format of the existing entries:

```
   1865 EQUS "Churchill (C)+"
```

Now put him into the part of the program which acts on the user's choice:

```
   675 CMP #ASC"C":BEQ Churchill
```

Next, a label and two lines to pass on to the main program the address of the *ChurchillData* which you are about to type in:

```
   734 .Churchill
   735 LDX #ChurchillData MOD &100
   736 LDY #ChurchillData DIV &100:RTS
```

In each case, if your addition is an exact match with the lines around it, then all is well. The precise line-number is not important, as long as your addition is clearly in the right area.

Now, to write the actual *ChurchillData*, you need to understand the following symbols:

:	before each phrase or punctuation-mark.
#	end of choice-group.
>	from here nothing goes to the printer.
+	print a blank line (i.e. carriage return).
*	end of 'poet'.
<	backspace (use before punctuation.)

It follows that the sequence :<<:text# means that the word "text" has a fifty-fifty chance of being printed, the first alternative << being nothing - just a couple of backspaces to tidy up the layout. I have used this trick quite a bit. Study the use of these symbols in the existing poets, and all should soon be clear.

THIS IS NOT THE BEGINNING OF THE END

Let's begin Churchill at the end of Wordsworth, with our label which we've already referred to:

```
  3081 .ChurchillData
```

Now a special line containing "&0C" to clear the screen (the only operation without a special symbol). Just copy exactly the first line of the other poets:

```
  3082 EQUS ":"+CHR$ &0C+"++#"
```

Now a choice of titles, perhaps?

```
  3083 EQUS ":CHURCHILL:WINNIE:VICTORY
 SPEECH#"
```

The program will pick one of the alternatives it finds between the previous hash and the one that ends this choice-group. Now we want a couple of blank lines:

```
  3084 EQUS ":++#"
```

In this last case there is only one item in the choice group - two plus signs to indicate carriage returns - so this item will always be chosen. Now for the fun bit:

```
  3085 EQUS ":We:I:You:They:Saddam
 Hussein:Yogi Bear#"
  3086 EQUS ":will:shall:must:can:are
 scared to#"
  3087 EQUS ":fight:struggle:run
 away:hide:do handstands#"
```

By now you will be worrying that there will soon be no spare line-numbers left: you can't write a new line 3090, as it would displace the old one. All you do is type RENUMBER. List the program again, and you will find that what was line 3087 a moment ago is now

```
  3150 EQUS ":fight:struggle.... etc.
```

Carry on as follows:

```
  3151 EQUS ":on:behind:at:under#"
```

etcetera, and RENUMBER whenever you run short of line-numbers. Try some alternative punctuation at the end of a sentence, e.g.

```
  3??? EQUS ":.:!#"
```

i.e. 'show a full stop or an exclamation mark.' When it's run, you'll find an ugly gap between the last word and the punctuation mark. This is because the poet inserts a space between all choices. Rewrite the line using the backspace symbol < to get rid of the gap:

```
  3??? EQUS ":<.:<!#"
```

At the end of the speech proper,put some carriage-returns and > to indicate 'printer should stop here':

```
  EQUS ":++++>#"
```

Now write a suitable final menu such as the others have, indicating that 'A' or 'Y' mean copy to the printer; 'N' means don't print, and 'Q' means quit. This menu can itself comprise many choices, as in Shakespeare, or consist of just one wording sandwiched between : and #. In Burns I have added a few bits of nonsense prefixed with :<< before the menu proper. Now end the whole thing:

```
  EQUS ":++*"
```

BUT IT IS THE END OF THE BEGINNING

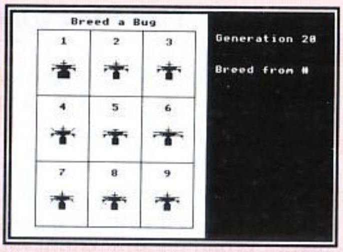
Make sure you've saved the final version, preferably on two separate discs, and run it. If it works first time, you're a genius; you'll probably need to do Break, then type OLD and keep checking your new data, especially the symbols ':', '#' and '*', until it works. Then comes an awful lot of tinkering to get rid of bad combinations.

Note that a choice group is not the same thing as a line of the program, though

Continued on page 46

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
THE EARTH FROM SPACE - draw a picture of the Earth as seen from any point in space
PERSONALISED ADDRESS BOOK - on-screen address and phone book
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.

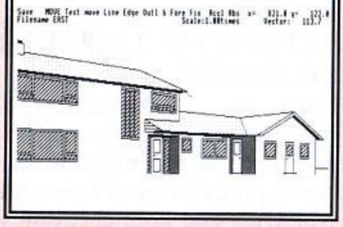
The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical 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

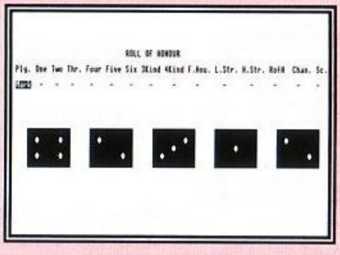
	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£5.95	**ASTAAD** (3.5" ADFS)	1408a	£5.95
Applications II (80 track DFS)	1411a	£4.00	**Applications II** (3.5" ADFS)	1412a	£4.00 ✓
Applications I Disc (40/80T DFS)	1404a	£4.00	**Applications I Disc** (3.5" ADFS)	1409a	£4.00 ✓
General Utilities Disc (40/80T DFS)	1405a	£4.00	**General Utilities Disc** (3.5" ADFS)	1413a	£4.00 ✓
Arcade Games (40/80 track DFS)	PAG1a	£5.95	**Arcade Games** (3.5" ADFS)	PAG2a	£5.95 ✓
Board Games (40/80 track DFS)	PBG1a	£5.95	**Board Games** (3.5" ADFS)	PBG2a	£5.95 ✓

All prices include VAT where appropriate. For p&p see Membership page.

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

Best of BEEBUG

Board Games


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

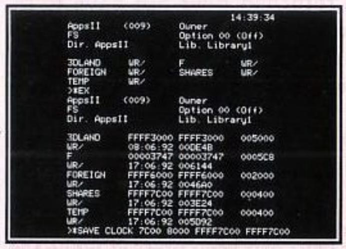
Cribbage - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

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.

Applications II Disc



```
                                  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
```

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

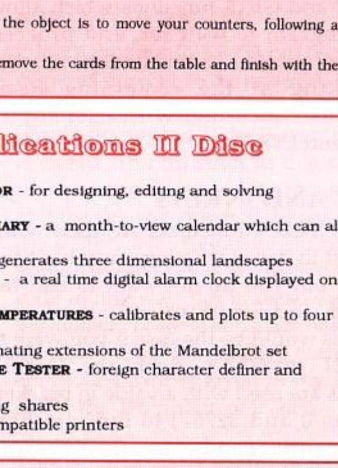
Julia Sets - fascinating extensions of the Mandelbrot set

Foreign Language Tester - foreign character definer and language tester

Share Investor - assists decision making when buying and selling shares

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

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.

	Stock Code	Price
File Handling for All Book	BK02b	£ 9.95
File Handling for All Disc (40/80T DFS)	BK05a	£ 4.75
Joint Offer book and disc (40/80T DFS)	BK04b	**£ 11.95**
Magscan (40 DFS)	0005a	£ 9.95
Magscan (80T DFS)	0006a	£ 9.95
Magscan (3.5" ADFS)	0007a	£ 9.95 ✓

	Stock Code	Price
File Handling for All Disc (3.5" ADFS)	BK07a	£ 4.75 ✓
Joint Offer book and disc (3.5" ADFS)	BK06b	**£ 11.95** ✓
Magscan Upgrade (40 DFS)	0011a	£ 4.75
Magscan Upgrade (80T DFS)	0010a	£ 4.75
Magscan Upgrade (3.5" ADFS)	1458a	£ 4.75

All prices include VAT where appropriate. For p&p see Membership page.

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

1st course

Input (3)

Alan Wrigley concludes his description of input processing in Basic.

So far in this series of articles I have described the use of the INPUT statement to obtain textual input from the user, and the GET and GET$ functions which allow your program to wait for, and process, single key presses. In this final part, I will be looking at the remaining Basic keywords concerned with user input: INKEY and INKEY$.

INKEY AND INKEY$

The two INKEY functions are used when you want to detect whether a key has been pressed, but you do not want to halt the program indefinitely until the event occurs, as would be the case if you were to use GET or GET$. Both the INKEY functions are used with a value in brackets (between 0 and 32767) to indicate how many centiseconds to wait before returning. If you specify zero, the call returns immediately. For example:

```
A%=INKEY(100)
```

would pause for a maximum of 1 second. I have said "a maximum of" because if a key is pressed before the time elapses, INKEY returns immediately with the ASCII value of the key pressed. If on the other hand the specified time elapses without a key being pressed, the function returns with a value of -1. In the case of INKEY$, the value returned is a string just like GET$, and if a key is not pressed then a null string is returned.

The BBC micro is designed in such a way that keypresses are stored in a keyboard buffer, from where they are removed in the correct order by commands which process input (or by flushing the buffer, as we will see in a moment). This enables you to type ahead regardless of what the program is doing, and is a feature which gave the BBC an advantage over many of its competitors at the time. Because of this feature, however, it is important to realise that there could already be keypresses waiting in the buffer before INKEY is called. In this case, the next character due to be removed from the buffer is detected by the function. In other words, the keypress returned does not have to be made during the actual time delay introduced by INKEY, but may have been pressed previously.

A typical example of the use of INKEY would be to implement a "hot keys" function. This would involve polling the keyboard at regular intervals to see if particular keys had been pressed while the program was running. For example, while printing a long document, you might want to give the user the option to stop the print job by pressing 'S'. It would be inconvenient to keep pausing to issue a prompt, so you might simply use INKEY(0) at the end of every line of print, as follows:

```
finished%=FALSE:REPEAT
PROCprintaline
A%=INKEY(0)
IF A%=83 OR A%=115 finished%=TRUE
UNTIL finished%
```

Here, *PROCprintaline* would print one line of the document, and would also set *finished%* to TRUE once the last line was printed. After each line, INKEY(0) would detect whether a key had been pressed. If the value returned was 83 or 115 it would mean the user had pressed the 'S' key at some point ('S' is ASCII 83, 's' is ASCII 115). *finished%* is therefore set to TRUE in order to end the job.

Because keys are stored in the buffer, there is a danger that INKEY might extract a key which was pressed some time ago, perhaps inadvertently. In the situation described above, the key might have been pressed before the program ever got to the section which prints the document. It is a good idea, therefore, to flush the keyboard buffer before entering the loop which processes the print job. This is done with the following command:

```
*FX 15,1
```

Bear in mind, too, that the user might press a key several times in one line, and only one of these presses would be removed from the buffer by INKEY. It might therefore be a good idea to flush the buffer at the start of each print line.

By and large, the same reasons apply for using INKEY in preference to INKEY$, and vice versa, as we saw last month with GET and GET$. However, if you want to return a string value and use the INSTR function to decode it as we did last month, you must be a little more careful. With GET$, the result can never be a null string, since the function will not return until a valid key is pressed. INKEY$, on the other hand, can return a null string as we described earlier, and if you then use this as the target string with INSTR, you will get a value of 1. In other words:

```
A%=INSTR("ABCD","")
```

will set A% to 1.

To go back to the print job example, suppose you want to offer the user a choice of hot keys - perhaps S to stop or P to pause (or their lower-case equivalents). A line such as:

```
ON (INSTR("SsPp",INKEY$(0))+1) DIV 2 PR
OCstop,PROCpause,PROCdonothing
```

will almost certainly result in the print job ending as soon as it has begun, because unless a key is pressed immediately, the first value returned by INKEY$ will be a null string. This will result in a value of 1 for the INSTR function, thus activating PROCstop. You would therefore need to alter the code as follows:

```
A$=INKEY$(0)
IF A$>"" ON (INSTR("SsPp",A$)+1) DIV 2
PROCstop,PROCpause,PROCdonothing
```

NEGATIVE INKEY

INKEY (but not INKEY$) can be used with a negative number in the brackets. This is used to detect whether a specific key is being physically pressed at the instant the function is called. The function in this case always returns immediately, with a value of TRUE (-1) if the key was down at the time, or FALSE (0) if not. The negative number which applies to each key is not related to its ASCII code in any way, but to its so-called *internal key number*. There is not space here to explain this, but a list of the negative numbers for the entire keyboard is given in the User Guide under the description of INKEY.

This is particularly useful in games, where movement of an object may be controlled by holding down a particular key. For example, you may want a spaceship to be moved left, right, up or down using the four cursor keys. The negative numbers for these are -26, -122, -58 and -42 respectively. Your program might then be based around a loop which, as well as moving other objects (aliens, bombs etc.), contains the following lines:

```
IF INKEY(-26) PROCleft
IF INKEY(-122) PROCright
IF INKEY(-58) PROCup
IF INKEY(-42) PROCdown
```

Each time round the loop, these lines scan the keyboard to see if any of the cursor keys are currently down, and call the appropriate procedure if this is the case.

We will conclude this article with a short listing which demonstrates the principle.

If you followed our earlier *1st Course* series on graphics programming (Vol.11 Nos. 4-7) you will remember the bouncing ball program from part 3 of the series. We will amend this so that the ball is moved under the control of the keyboard, using 'Z' and 'X' for horizontal movement, and '*' and '?' for vertical, in common with most games. The listing also demonstrates the use of INKEY(0) - by pressing the C key, the colour of the ball will change.

```
 10 MODE0
 20 VDU23,240,0,0,7,31,63,127,255,255
 30 VDU23,241,0,0,224,248,252,254,255,255
 40 VDU23,242,255,255,127,63,31,7,0,0
 50 VDU23,243,255,255,254,252,248,224,0,0
 60 a$=CHR$240+CHR$241+CHR$10+CHR$8+CHR$8
+CHR$242+CHR$243
 70 VDU5:step%=8:col%=1
 80 VDU19,1,col%,0,0,0
 90 x%=400:y%=400:GCOL4,1
100 REPEAT
110 IF (INKEY(0) OR 32)=99 PROCcol
120 IF INKEY(-98) x%=x%-step%
130 IF INKEY(-67) x%=x%+step%
140 IF INKEY(-73) y%=y%+step%
150 IF INKEY(-105) y%=y%-step%
160 IF x%<0 x%=0 ELSE IF x%>1248
x%=1248
170 IF y%<64 y%=64 ELSE IF y%>1024
y%=1024
180 PROCmove
190 UNTIL FALSE
200 DEF PROCcol
210 col%=col% EOR 3
220 VDU19,1,col%,0,0,0
230 ENDPROC
240 DEF PROCmove
250 MOVE x%,y%:PRINTa$
260 *FX19
270 MOVE x%,y%:PRINTa$
280 ENDPROC
```

The ball is made up by re-defining four characters as before (refer to Vol.11 No. 6, page 24 for a fuller description). However, instead of moving the ball automatically, the main program loop (lines 100-190) now tests for keys pressed. First of all, line 110 uses INKEY(0) to see if C has been pressed, and if so it calls *PROCcolour* to change the ball colour from red to green or vice versa. Then the program uses the negative INKEY function to check for each of the movement keys in turn. If any of them is down at the time, the appropriate screen co-ordinates will be updated and the ball redrawn with *PROCmove*. Lines 160-170 ensure that the ball cannot move outside the screen area. If you release all the keys, the ball will stop moving.

Note that *PROCmove* is called whether or not a key is being pressed - this is because the effect of the procedure is to draw and then immediately erase the ball (using GCOL action 4) so that if it subsequently moves its position, the background will have been restored. If *PROCmove* were only called when a key is pressed, the ball would be invisible at other times.

You will find that you can move the ball diagonally by holding down a horizontal and a vertical movement key at the same time. The diagonal movement is actually an illusion - what happens is that a horizontal movement is processed followed immediately by a vertical movement. You can also press C to change the colour while you are holding down one of the movement keys, but not if you are holding down two other keys. The reason for this is too complex to be discussed in this article.

This concludes our look at the subject of input, and next month *1st Course* will move on to a fresh topic.

A Fast Single Drive ADFS Backup (Part 2)

Roger Smith completes his ADFS backup utility.

Last month's article presented the program *MakeBackup* which produces *FastBackup*, a machine code program for performing fast backups of ADFS discs. Two programs are listed at the end of this month's article: *InitBackup*, which creates a backup disc for a specific original disc, and *CheckADFS*, which compares two ADFS discs.

USING THE PROGRAMS

Both programs should be entered and saved to disc. When InitBackup is run it searches for FastBackup (it looks in the current directory, then the root directory and finally in directory "$.Lib*"), and then asks for the source disc to be inserted. After reading the disc title it will offer to write FastBackup to the disc if it is not already present; it then asks for the backup disc. After checking that the backup disc is the same size as the source disc, it writes an encoded version of the source disc title onto the backup disc. If the source disc has not been given a title, InitBackup will prompt for a new title - try to keep disc titles unique as this is how FastBackup checks that the correct backup disc is being used. The backup disc supplied to InitBackup must be formatted, but its contents do not matter.

For safety, try out InitBackup on discs which do not matter, or write protect the source disc.

The CheckADFS program compares two ADFS discs. It displays the free space map of the first disc and then reads all the sectors of the disc and stores a 16-bit cyclic redundancy check (CRC) for each sector. It then reads all the sectors of the second disc, comparing CRC values as it reads each sector. It displays a map of those sectors where the sector CRCs disagree. When FastBackup has been used to copy a disc there will be a discrepancy in sector 6 (because the disc title has been altered) and there may be discrepancies in the free space sectors. Any other differences indicate a fault in the copying process. Due to the nature of CRCs, there is a small possibility (1 in 2^16) that two different sectors will be shown as equivalent; this should not cause a problem in practice. CheckADFS is a useful way of verifying the correct operation of any backup utility - the last thing you want is backups which are faulty! I have used FastBackup in its current form for eight months without any problems and now only check its operation intermittently.

PROGRAM NOTES

InitBackup makes extensive use of OSFILE calls, which are made easier to use with the function FNosfile. It also uses OSWORD &72 to directly read and write ADFS sectors; the Basic procedures *PROCget* and *PROCput* do this. The coding of these procedures makes use of the (little known) fact that the DEF keyword comments out the rest of the line when that line is executed, so when PROCget is called, line 1440 is ignored and execution continues with line 1450.

CheckADFS reads in a number of disc sectors at a time, determined by the value in the variable *chunk*, and calculates the CRC for each sector using the machine code routine generated by *PROCmc*. The CRC algorithm used is the same as that used by the cassette filing system. The size of *chunk* (line 110) is a suitable value for use on a Model B; if memory permits, it can be increased to 64 with a slight increase in speed. I have found that further increasing it to 128 slows the program down as the processing of the sectors takes longer than the disc time-out, and the drive motor stops and starts between each batch of sectors.

I hope you find these programs useful, they certainly take a lot of the donkey work out of backing up ADFS discs.

```
   10 REM Program CheckADFS
   20 REM Version B 1.00
   30 REM Author  Roger Smith
   40 REM BEEBUG  April 1993
   50 REM Program Subject to Copyright
   60 :
  100 MODE 7
  110 chunk=32:OSWORD=&FFF1:MaxSize=2560
  120 DIM track chunk*256,pblk 15
  130 DIM crcl MaxSize*2
  140 PROCmc
  150 :
  160 REPEAT
  170 CLS
  180 PRINT"ADFS disc comparison"
  190 PRINT''"Insert reference disc and h
it a key"
  200 A%=GET:PRINT
  210 Nsect=1:PROCreadchunk(0):DiscSize=
track!252 AND &FFFFFF
  220 Nsect=DiscSize
  230 IF Nsect>MaxSize THEN PRINT"Too ma
ny sectors"'"Increase MaxSize to ";Nsect
:END
  240 VDU 14
  250 *mount
  260 *map
  270 FOR s%=0 TO Nsect-1 STEP chunk:PRO
Creadchunk(s%):PROCsaveCRCs(s%):NEXT
  280 PRINT'"Insert copy disc and hit a
key";
  290 A%=GET:PRINT''"Differences"'"Addre
ss    Length"
  300 Nsect=1:PROCreadchunk(0):Nsect=tra
ck!252 AND &FFFFFF
  310 IF Nsect<>DiscSize THEN PRINT"Disc
s are different sizes":END
  320 start=-1
  330 FOR s%=0 TO Nsect-1 STEP chunk:PRO
Creadchunk(s%):PROCcheck(s%):NEXT
  340 PROCsame(Nsect)
  350 PRINT"Press any key...";A%=GET
  360 UNTIL FALSE
  370 END
  380 :
 1000 DEF PROCmc
 1010 DIM mc_area 100
 1020 addr=&70
 1030 crc=&72
 1040 H=crc:L=crc+1
 1050 FOR J%=0 TO 3
 1060 P%=mc_area
 1070 [OPT J%
 1080 .docrc
 1090 LDA #0:STA H:STA L:TAY
 1100 .nbyt LDA H:EOR (addr),Y:STA H:LDX
#8
 1110 .loop LDA H:ROL A:BCC b7z
 1120 LDA H:EOR #8:STA H:LDA L:EOR #&10:
STA L
 1130 .b7z ROL L:ROL H:DEX:BNE loop
 1140 INY:BNE nbyt
 1150 RTS
 1160 ]
 1170 NEXT
 1180 ENDPROC
 1190 :
 1200 DEF PROCreadchunk(N%)
 1210 len=chunk
 1220 IF (N%+len)>Nsect THENlen=Nsect-N%
 1230 PRINT"Reading sectors ";N%;"-";N%+
len-1;
 1240 pblk!1=track
 1250 sector=N%
 1260 pblk?5=8
```

```
1270 pblk?6=sector DIV &10000
1280 pblk?7=(sector DIV &100) AND &FF
1290 pblk?8=sector AND &FF
1300 pblk?9=len
1310 pblk?10=0
1320 pblk!11=0
1330 A%=&72:X%=pblk AND &FF:Y%=pblk DIV
&100
1340 CALL OSWORD
1350 VDU13:PRINTSPC(39);:VDU13
1360 ENDPROC
1370 :
1380 DEF PROCsaveCRCs(N%)
1390 addr?0=track MOD 256:addr?1=track
DIV 256
1400 crcptr=crc1+N%*2
1410 FOR J%=0 TO len-1
1420 CALL docrc:crcptr?0=crc?0:crcptr?1
=crc?1
1430 crcptr=crcptr+2:addr?1=addr?1+1
1440 NEXT
1450 ENDPROC
1460 :
1470 DEF PROCcheck(N%)
1480 addr?0=track MOD 256:addr?1=track
DIV 256
1490 crcptr=crc1+N%*2
1500 FOR S%=N% TO N%+len-1
1510 CALL docrc
1520 IF crcptr?0=crc?0 AND crcptr?1=crc
?1 THEN PROCsame(S%) ELSE PROCdiff(S%)
1530 crcptr=crcptr+2:addr?1=addr?1+1
1540 NEXT
1550 ENDPROC
1560 :
1570 DEF PROCsame(S%)
1580 IF start<0 ENDPROC
1590 PRINTRIGHT$("00000"+STR$~start,6);
1600 PRINT"  :  ";
1610 PRINTRIGHT$("00000"+STR$~(S%-start
),6)
1620 start=-1
1630 ENDPROC
1640 :
1650 DEF PROCdiff(S%)
1660 IF start=-1 THEN start=S%
1670 ENDPROC
```

```
  10 REM Program InitBackup
  20 REM Version B 1.00
  30 REM Author  Roger Smith
  40 REM BEEBUG  April 1993
  50 REM Program Subject to Copyright
  60 :
 100 MODE 7
 110 DIM blk &11,misc 255,sect6 255
 120 name$=STRING$(20,"*")
 130 OSARGS = &FFDA:OSFILE = &FFDD
 140 OSWORD = &FFF1
 150 A%=0:Y%=0:F%=USR(OSARGS) AND 255
 160 IF F%<>8 THEN PRINT"ERROR - not us
ing ADFS":STOP
 170 PRINT''"Searching for FastBackup c
ode...";
 180 Code$=FNfindcode:PRINT
 190 IF Code$="" THEN PRINT"ERROR - can
not find FastBackup code":STOP
 200 T%=FNosfile(5,Code$,0,0,0,0)
 210 CodeLen=blk!&0A:CodeLAD=blk!&02
 220 CodeXAD=blk!&06:DIM code CodeLen
 230 T%=FNosfile(&FF,Code$,code,0,0,0)
 240 T$=CHR$141+CHR$131+"      --- InitB
ackup ---"
 250 CLS:PRINT''T$'T$''"This program cr
eates a backup disc for"'"use with the F
astBackup utility."''"Both the SOURCE an
d BACKUP discs must"'"be formatted."
 260 REPEAT
 270 PRINT'''"Insert";CHR$130;"SOURCE";
CHR$135;"disc and hit a key";
 280 Z%=GET:PRINT:*MOUNT
 290 REPEAT
 300 name$="":term=-1
 310 PROCget(0,misc)
 320 size0=misc!252 AND &FFFFFF
 330 PROCget(6,sect6)
 340 J%=0
 350 REPEAT
 360 C%=sect6?(&D9+J%)
 370 IF C%>31 THEN name$=name$+CHR$C% E
LSE term=C%
 380 J%=J%+1
 390 UNTIL J%>18 OR term>=0
 400 IF name$="$" THEN PROCrename
 410 UNTIL name$<>"$"
```

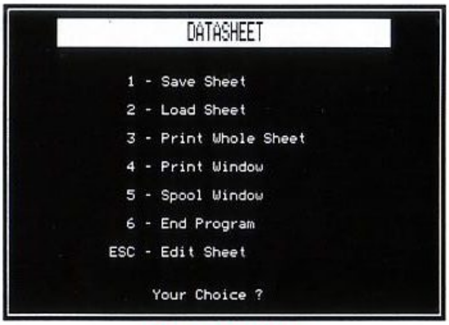
```
  420 PRINT"Disc is called";CHR$134;name
$
  430 IF term<>13 THEN PRINT;CHR$129;"
(probably a BACKUP)"
  440 IF term=13 AND FNfindcode="" THEN
PROCwritecode
  450 PRINT''"Insert";CHR$129;"BACKUP";CH
R$135;"disc and hit a key";
  460 Z%=GET:PRINT
  470 PROCget(0,misc)
  480 size1=misc!252 AND &FFFFFF
  490 IF size0<>size1 THEN PRINT"Wrong s
ize BACKUP disc!":STOP
  500 PROCget(6,misc)
  510 FOR J%=0 TO 18
  520 misc?(&D9+J%) = sect6?(&D9+J%) EOR
 15
  530 NEXT
  540 PROCput(6,misc)
  550 PRINT''"Initialise another backup?"
;
  560 UNTIL (GET AND &DF)<>89
  570 PRINT:END
  580 :
 1000 DEF PROCrename
 1010 LOCAL X%,Y%
 1020 PRINT"This disc is called";CHR$134
;"$";CHR$7
 1030 PRINT"Please supply a new title:"
 1040 $misc="TITLE "
 1050 INPUT LINE ">>> " $(misc+6)
 1060 X%=misc MOD 256:Y%=misc DIV 256:CA
LL &FFF7
 1070 ENDPROC
 1080 :
 1090 DEF FNfindcode
 1100 T%=FNosfile(5,"$.Lib*",0,0,0,0)
 1110 LIB%=T%=2
 1120 T%=FNosfile(5,"FastBackup",0,0,0,0
)
 1130 IF T%=1 THEN ="FastBackup"
 1140 T%=FNosfile(5,"$.FastBackup",0,0,0
,0)
 1150 IF T%=1 THEN ="$.FastBackup"
 1160 IF NOT LIB% THEN =""
 1170 T%=FNosfile(5,"$.Lib*.FastBackup",
0,0,0,0)
 1180 IF T%=1 THEN ="$.Lib*.FastBackup"
 1190 =""
 1200 :
 1210 DEF PROCwritecode
 1220 PRINT"Write FastBackup code to thi
s disc?";
 1230 Z%=GET AND &DF:PRINT:IF Z%<>89 THE
N ENDPROC
 1240 T%=FNosfile(5,"Lib*",0,0,0,0)
 1250 IF T%=2 THEN PRINT"Put it in libra
ry?";:Z%=GET AND &DF:PRINT ELSE Z%=0
 1260 name$="$.FastBackup"
 1270 IF Z%=89 THEN name$="$.Lib*.FastBa
ckup"
 1280 T%=FNosfile(0,name$,CodeLAD,CodeXA
D,code,code+CodeLen)
 1290 ENDPROC
 1300 :
 1310 DEF FNosfile(A%,name$,LAD,XAD,SAD,
EAD)
 1320 LOCAL X%,Y%
 1330 X%=blk MOD 256:Y%=blk DIV 256
 1340 $misc=name$
 1350 blk?&00=misc MOD 256
 1360 blk?&01=misc DIV 256
 1370 blk!&02=LAD
 1380 blk!&06=XAD
 1390 blk!&0A=SAD
 1400 blk!&0E=EAD
 1410 =USR(OSFILE) AND 255
 1420 :
 1430 DEF PROCget(S%,mem):blk?&05=&08
 1440 DEF PROCput(S%,mem):blk?&05=&0A
 1450 LOCAL A%,X%,Y%
 1460 A%=&72
 1470 X%=blk MOD 256:Y%=blk DIV 256
 1480 blk?&00 = 0
 1490 blk!&01 = mem
 1500 REM blk?&05 is function code
 1510 blk?&06 =(S% DIV &10000)AND &1F
 1520 blk?&07 = S% DIV &100
 1530 blk?&08 = S%
 1540 blk?&09 = 1 : REM one sector
 1550 blk?&0A = 0
 1560 blk!&0B = 0
 1570 CALL OSWORD
 1580 IF ?blk=0 THEN ENDPROC
 1590 PRINT''"Disc error: &";~?blk
 1600 STOP
```

Datasheet2

Stephen Colebourne upgrades his spreadsheet program.

The original Datasheet was published in BEEBUG Vol. 11 Nos. 1 & 2. The listing below introduces three improvements to the spreadsheet program. These are:

1. Minor correction ensuring that numbers are rounded up correctly, e.g. 1.175 is shown as 1.18 not 1.17 after the correction.
2. Major improvement to formulae calculations, e.g. Average, Minimum, Maximum and VAT calculations.
3. Substantial improvement to the printer handling abilities, i.e. four pre-selected output styles/sizes are offered not one.



```
                DATASHEET

          1 - Save Sheet
          2 - Load Sheet
          3 - Print Whole Sheet
          4 - Print Window
          5 - Spool Window
          6 - End Program
        ESC - Edit Sheet

             Your Choice ?
```

The main menu

Load the original program, delete lines 2000 to 2130 inclusive and add the lines listed below. The new version of the entire program is on this month's disc as *DSHEET+*.

Using the new version the improvements should become apparent. Each time part of the sheet is to be printed you are asked to select from Pica, Elite, Condensed Pica and Condensed Elite (Set up on Star LC24-200). These choices can be changed in data statements at the end of the program.

The other new commands are formulae commands. Enter some numbers, then press f0 for a formula. Try any of the following, press COPY to evaluate:

S(AA,AD)	- Sum from AA to AD inclusive
A(AA,AE)	- Mean average AA to AE
M(CB,CH)	- Maximum of CB to CH
N(AF,GF)	- Minimum of AF to GF
V(DC)	- Gives 17.5% of DC
W(BF)	- Gives BF + 17.5%
X(CD)	- Gives CD ex VAT

Note that the VAT rate can be changed in *PROCUSER*, and that *PROCSUM* is not new (although it now works differently). Why not try adding your own functions?

```
   10 REM Program Datasheet2
   40 REM BEEBUG  April 1993
  790 V=VAL(C$)
 1000 IFPX$="S" IFPY$="(" IFC%=0 E%=5:PR
OCFUNC(2,"SUM")
 1001 IFPX$="A" IFPY$="(" IFC%=0 E%=5:PR
OCFUNC(2,"AVGE")
 1002 IFPX$="M" IFPY$="(" IFC%=0 E%=5:PR
OCFUNC(2,"MAX")
 1003 IFPX$="N" IFPY$="(" IFC%=0 E%=5:PR
OCFUNC(2,"MIN")
 1004 IFPX$="V" IFPY$="(" IFC%=0 E%=5:PR
OCFUNC(1,"VAT")
 1005 IFPX$="W" IFPY$="(" IFC%=0 E%=5:PR
OCFUNC(1,"WVAT")
 1006 IFPX$="X" IFPY$="(" IFC%=0 E%=5:PR
OCFUNC(1,"XVAT")
 1500 DEF PROCFUNC(A%,C$)
 1510 IFMID$(F$,P%+A%*3+1,1)<>")" PROCFC
D(1):ENDPROC
 1520 IFFNPAIR(MID$(F$,P%+A%*3-1,2)) ORL
EN(E$)>235 PROCFCD(8):ENDPROC
 1530 IFA%=1 PROCFUNC2:ENDPROC
 1540 T%=PX%:U%=PY%
 1550 IFFNPAIR(MID$(F$,P%+2,2)) PROCFCD(
8):ENDPROC
 1560 T%=T%-PX%:U%=U%-PY%
 1570 IFNOT(T%=0EORU%=0) PROCFCD(8):ENDP
ROC
 1580 E$=E$+"FN"+C$+"("+STR$(PX%)+","+ST
R$(PY%)+","+STR$(T%)+","+STR$(U%)+")"
 1590 P%=P%+7
 1600 ENDPROC
 1610 :
 1620 DEF PROCFUNC2
 1630 E$=E$+"FN"+C$+"("+STR$(PX%)+","+ST
R$(PY%)+")"
 1640 P%=P%+4
 1650 ENDPROC
 1660 :
 1670 :
 1680 DEF FNVAT(A%,B%)
 1690 =D(A%,B%)*(VAT/100)
 1700 :
 1710 DEF FNWVAT(A%,B%)
 1720 =D(A%,B%)*(1+VAT/100)
 1730 :
 1740 DEF FNXVAT(A%,B%)
 1750 =D(A%,B%)/(1+VAT/100)
 1760 :
 1770 DEF FNMAX(A%,B%,C%,D%)
 1780 V=D(A%,B%)
 1790 FORZ%=1TOABS(C%+D%)
 1800 A%=A%+SGN(C%):B%=B%+SGN(D%)
 1810 IFD(A%,B%)>V V=D(A%,B%)
 1820 NEXT
 1830 =V
 1840 :
 1850 DEF FNMIN(A%,B%,C%,D%)
 1860 V=D(A%,B%)
 1870 FORZ%=1TOABS(C%+D%)
 1880 A%=A%+SGN(C%):B%=B%+SGN(D%)
 1890 IFD(A%,B%)<V V=D(A%,B%)
 1900 NEXT
 1910 =V
 1920 :
 1930 DEF FNSUM(A%,B%,C%,D%)
 1940 V=D(A%,B%)
 1950 FORZ%=1TOABS(C%+D%)
 1960 A%=A%+SGN(C%):B%=B%+SGN(D%)
 1970 V=V+D(A%,B%)
 1980 NEXT
 1990 =V
 2000 :
 2010 DEF FNAVGE(A%,B%,C%,D%)
 2020 =FNSUM(A%,B%,C%,D%)/(ABS(C%+D%)+1)
 2030 :
 3520 IFFNPCODES VDU26:ENDPROC
 3590 IFE% VDU1,27,1,64,3:PC%=COLS%DIVCW
%-1:ELSEOSCLI("SPOOL")
 3800 DEF FNPCODES
 3810 IFE%=0 PROCSPOOL:=0
 3820 CLS:FORZ%=0TO3
 3830 PRINTTAB(6,3+Z%*2);CHR$134;CD$(Z%)
;
 3840 NEXT
 3850 PRINTTAB(11,13);CHR$131;"Your Choi
ce ";:INPUT"? "CD%
 3860 IFCD%<1 ORCD%>4 THEN=1
 3870 VDU2:FORZ%=1TO10
 3880 VDU1,CD%(CD%-1,Z%)
 3890 NEXT
 3900 PC%=CD%(CD%-1,0)DIVCW%-1
 3910 =0
 3920 :
 5190 PRINT,D(V%,W%)+0.000001;
 5270 @%=AT1%:PRINTRIGHT$(S$+STR$(D(X%,Y
%)+0.000001),CW%);
 5955 DIMCD$(3),CD%(3,10):CD%=0
 6052 FORCD%=0TO3:READCD$(CD%)
 6054 FORZ%=0TO10:READCD%(CD%,Z%)
 6056 NEXT,
 6252 DATA1 - Pica (10 cpi),72,27,80,0,0
,0,0,0,0,0,0
 6253 DATA2 - Elite (12 cpi),88,27,77,0,
0,0,0,0,0,0,0
 6254 DATA3 - Condensed Pica (17 cpi),12
8,27,80,15,0,0,0,0,0,0,0
 6255 DATA4 - Condensed Elite (20 cpi),1
44,27,77,15,0,0,0,0,0,0,0
 6320 REM**No.of columns when spooled**
 6332 REM**VAT percentage**
 6334 VAT=17.5
```

Spriter (Part 2)

Alan Blundell concludes his examination of Acorn's undocumented graphics ROM image for the Master.

Last issue, we covered the general functions of Spriter and looked at how sprites are defined, together with the housekeeping functions of the ROM image like sprite load/save, etc. In this article, I will cover the basics of how to use Spriter in your own programs to display and animate sprites.

BUT FIRST, THIS

Before going on to that here is a summary of the commands obeyed by Spriter which weren't covered last time.

***SGET** is an easy way to define a sprite, if the image you want to use as a sprite is visible on the screen, or can be displayed on the screen.

***SGET (*n*)**, where *n* is a sprite number, will capture an image from screen memory and store it in sprite number *n*. It is actually very easy to use: before calling *SGET, you need to perform two Basic MOVE commands to set two diagonally opposite corners of the rectangular part of the screen you wish to make into a sprite. Having done that, *SGET (*n*) does the rest.

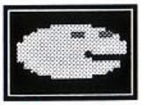
***SCHOOSE** selects a sprite for plotting on the screen - you need to select the sprite, as there is no one command to both select and display a sprite. Both *SGET and *SCHOOSE have alternative means of being called. If a star command doesn't suit the way your program works, the same results can be achieved via VDU commands. This works as an extension of the VDU23 command, commonly used to redefine characters.

VDU 23,27,1,n,0;0;0; is the equivalent of *SGET. Note that you still need to have performed the two MOVE commands to select the area of the screen to be read.

VDU 23,27,0,n,0;0;0; is the equivalent of *SCHOOSE. For this command no setting up is needed, as it merely selects an existing sprite for the next display command.

The actual display of a sprite can also be achieved by a VDU command, this time by a PLOT (VDU25) command. The plotting of sprites is handled in the same way as any other graphics plotting, such as lines, circles, triangles, etc. The PLOT commands relevant to sprites are codes &E8 to &EF. Acorn allocated PLOT codes in blocks of 8, so that there are 8 ways of plotting a line, 8 ways of plotting a triangle, and so on. Only the relative and absolute co-ordinates part of these options is really useful with sprites; Normally, you would use n+1 (233) for relative plotting and n+5 (237) for absolute plotting. You'll find a complete list of PLOT codes on page 231 of the Welcome Guide.

EXAMPLES


The sprite created by Sprite1

That essentially gives you all the information you need to use Spriter for yourself. However, it is always easier to get the feel of how to do something by seeing an example of how it all works in practice, so *Sprite1* is an example of just that. The sprite it uses isn't exactly original and the program only allows you to move it around a blank screen, but that is all to the good: the whole point of the program is to show how the sprite commands work, so a complex program would defeat the object of the exercise. Because I couldn't provide a sprite directly as part of the listing, much of the program is devoted to defining a few character graphics and then using *SGET to grab these into a sprite.

EXPLANATION OF SPRITE1

Lines 80-140 and 480-520 redefine a few characters and display these on the screen as a two-coloured 'Pacman' type creation.

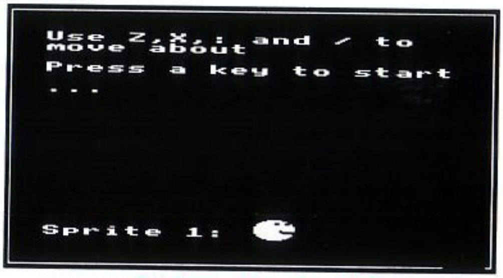
Lines 230-250 use the VDU code equivalent of *SGET to grab Pacman as a sprite. The two MOVE commands mark two diagonally opposite corners of a notional box around the sprite.

Line 300 selects GCOL type 3 - EOR plotting of graphics. The graphics colour chosen does not matter, as sprites by their nature contain their own colour information. EOR plotting is usually the best option if graphics are to be animated: the first plot displays the sprite, a second plot at the same position will EOR the displayed sprite with itself, which is the quickest way of removing it from the display before re-plotting it at its new position.

Line 310 uses the VDU equivalent of *SCHOOSE to select sprite number 1 (defined in line 250) to be the next sprite displayed. This command stays in effect until a new *SCHOOSE command is issued - in other words, the selected sprite stays selected until another selection is made.

Line 340 Plots the sprite on the middle of the screen (PLOT 237 = &E8 + 5 to PLOT to absolute screen co-ordinates).

Finally, lines 350-440 form an endless loop which checks for keypresses for up/down/left/right, keeps track of the position in X% and Y%, and (in line 430) moves the sprite. The movement works by plotting the sprite at its original position for a second time (using EOR mode) to remove it from the screen, then plotting it again in its new position. Note the use of *FX19 (wait for vertical sync.) to reduce flicker.



```
Use Z,X,: and / to
move about
Press a key to start
...


Sprite 1:
```

The screen from Sprite1

MORE SOPHISTICATED ANIMATION

If you try Sprite1, you will probably see a fair bit of flicker as the sprite is moved about the screen. Much of this is due to the slowness of Basic for this sort of purpose. The actual sprite plotting routines in Spriter are quite speedy, but for each move, Basic has to interpret the VDU commands and pass them on to the MOS before they reach Spriter. Also, don't forget, all this has to be done twice for each step in the movement of the sprite; once to delete it from its original position, once to plot it in the new position.

ALL THIS AND SPRITE2

Listing 2 demonstrates one of the ways round this, by halving the number of sprite plots needed for animation, so doubling the speed. The method relies on creating a second sprite which consists of the original EORed with itself at a pre-determined offset position. In this simple example, only horizontal movement of the sprite from left to right is dealt with but the principle can be extended. This example use the *SGET and *SCHOOSE star commands, so that you can see the alternative way of doing things. Also, it uses relative PLOT coordinates with PLOT code &E8+1 (233), rather than the absolute co-ordinates used in the previous example.

The program simply alternates between two methods of displaying a moving sprite - the first being the 'delete and display in new position' two step method, the second being the complex sprite approach. Because the sprite used contains a copy of the original 'pre-EORed' with itself, only one step is needed to both delete the original and display in the new position. In this case, the new position is 16 absolute graphical units to the right (see lines 210-250). If you type in and run this listing, you are first asked for 5 characters of input text. This is used to create a relatively large sprite, which emphasizes any flicker in the display. Then you should be able to see clearly the improvement in display which can be achieved with a bit of effort.

OVER TO YOU

I hope that the brief guide to the mysteries of Spriter in this and the previous article has been enough to show you how to use it in your own programs. Perhaps we'll see more on the subject in future issues, or even programs published which make use of the 'new' facilities.

Listing 1

```
   10 REM Program Sprite1
   20 REM Version B 1.0
   30 REM Author  Alan Blundell
   40 REM BEEBUG  April 1993
   50 REM Program subject to copyright
   60 MODE 2
   70 *SPRITE ON
   80 FOR loop1%=251 TO 255
   90 VDU23,loop1%
  100 FOR loop2%=0 TO 7
  110 READ byte%
  120 VDU byte%
  130 NEXT
  140 NEXT
  150 PRINT TAB(0,24)"Sprite 1:"
  160 VDU5
  170 GCOL 0,3
  180 MOVE 670,280
  190 VDU251,252,10,8,8,253,254,8,11
  200 GCOL 3,2
  210 VDU255
  220 VDU 4
  230 MOVE 650,200
  240 MOVE 800,300
  250 VDU 23,27,1,1,0;0;0;
  260 :
  270 PRINT TAB(0,5)"Use Z,X,: and / to"
'"move about"'
  280 PRINT"Press a key to start"'"...";
  290 IF GET CLS
  300 GCOL 3,0 :REM The colour doesn't m
atter - only the EOR option!
  310 VDU 23,27,0,1,0;0;0;
  320 X%=600
  330 Y%=600
  340 PLOT 237,X%,Y%
  350 REPEAT
  360 H%=0
  370 V%=0
  380 IF INKEY(-98) THEN H%=-8   :REM Z
  390 IF INKEY(-67) THEN H%=8    :REM X
  400 IF INKEY(-105) THEN V%=-8  :REM /
  410 IF INKEY(-73) THEN V%=8    :REM :
  420 *FX19
  430 IF H%<>0 OR V%<>0 THEN PLOT 237,X%
,Y%:X%=X%+H%:Y%=Y%+V%:PLOT 237,X%,Y%
  440 UNTIL 0
  450 END
  460 :
  470 REM Character definition data
  480 DATA 7,31,63,127,127,255,255,255
  490 DATA 224,248,252,254,254,255,255,2
55
  500 DATA 255,255,255,127,127,63,31,7
  510 DATA 223,192,224,254,254,252,248,2
24
  520 DATA 0,0,204,204,0,0,0,0
```

Listing 2

```
  10 REM Program Sprite2
  20 REM Version B 1.0
  30 REM Author  Alan Blundell
  40 REM BEEBUG  April 1993
  50 REM Program subject to copyright
  60 MODE 2
  70 *SPRITE ON
  80 PRINT''"Type 5 letters then";
  90 PRINT''"press RETURN: ";
 100 COLOUR 1
 110 INPUT""sprite$
 120 MOVE 880,900
 130 MOVE 1200,860
 140 *SGET 0
 150 GCOL 3,0
 160 *SCHOOSE 0
 170 COLOUR 2
 180 PRINT TAB(0,25)"Sprite 0:"
 190 PLOT 237,650,190
 200 PRINT TAB(0,27)"Sprite 1:"
 210 PLOT 237,650,120
 220 PLOT 237,666,120
 230 MOVE 650,120
 240 MOVE 1000,180
 250 *SGET 1
 260 choice=-1
 270 REPEAT
 280 choice = NOT choice
 290 IF choice PROCcomplex ELSE PROCsim
ple
 300 UNTIL 0
 310 END
 320 :
 330 REM Simple EOR method
 340 DEF PROCsimple
 350 PRINT TAB(0,8)"Simple sprite :"
 360 *SCHOOSE 0
 370 MOVE 0,600
 380 FOR loop%=0 TO 55
 390 *FX 19
 400 PLOT 233,16,0
 410 *FX 19
 420 PLOT 233,0,0
 430 NEXT
 440 ENDPROC
 450 :
 460 REM Complex EOR method
 470 DEF PROCcomplex
 480 PRINT TAB(0,8)"Complex sprite:"
 490 *SCHOOSE 0
 500 PLOT 237,16,600
 510 MOVE 0,600
 520 *SCHOOSE 1
 530 FOR loop%=0 TO 55
 540 *FX 19
 550 PLOT 233,16,0
 560 NEXT
 570 *SCHOOSE 0
 580 PLOT 233,16,0
 590 ENDPROC
```

The Sideways Poet (continued from page 31)

for readability I have mostly made it so here. As this assembly language program assembles the machine code, all the line numbers and EQUS directives disappear; they were only there to provide a structure for the programmer. You could in principle write the whole set of data as one long line, except that the Beeb can't handle lines of more than about 250 characters, but you can tag short groups such as :++* on the end of the previous line, or split a very long group into two lines. Each choice group should contain a maximum of 15 alternatives: if you add more, the extras will never be chosen. If you really need to, you can extend this limit to 31 by changing the statement AND #&0F in line 1240 to AND #&1F, but at some cost in speed.

Here are a few ideas for suitable new 'poets': The Lambton Worm; Jerusalem; a script for East Enders, Neighbours or The Archers; Horatius at the Bridge; a worn-out proverb; Jabberwocky. I hope you have some fun. Always bear in mind the immortal words of Hermann Goering: "Wenn ich das Wort 'Kultur' hoere, greife ich nach meinem21 Revolver." (When I hear the word 'culture', I reach for my revolver.)

Machine Code Corner

Mr Toad goes rummaging about in the Basic ROM.

Happy Spring, bipeds. Mr T has just come back from the World Hamster Fondling Olympiad in Bognor, and let me tell you, that final round was pretty close. The chimney of the old Beeb fairly belched sparks and smoke as it added up the scores.

Despite severe exhaustion, Mr T has been labouring day and night since his return to prepare Part 2 of his epic answer to one Silas Brown. The question remaining is: "How do you access Basic commands from machine-code?"

Well, generally you don't, in the strict sense. Your code can switch in the Basic ROM, as explained last time, with LDA #&0C:STA &FE30, unless the code is designed to be running with Basic, anyhow. You can then do a JSR to any subroutine in the Basic ROM - i.e. one known to end with an RTS - *if* you've got a disassembly of the ROM. Such routines may or may not be in the same place in the various versions of Basic: very likely not, so compatibility with other machines would be doubtful. Certainly, everything in Basic has moved in the new bolt-on OS chip, and some commercial software which uses Basic's code has had to be updated for the new ROM. Many hobbyist programmers are sure to have done it, and the addresses of some such routines are sure to have appeared in various publications over the years, but it's not normally worth the trouble, in Mr T's view, except perhaps in the case of a few especially useful bits of code. Mr T loathes disassembling and has never tried to access the Basic ROM. Why can't we have the Beeb equivalent of that wonderful book, "The Complete Spectrum Rom Disassembly?" In fact, I believe one was published early on, but I have never actually seen it.

The bits of code which really would be useful, to Mr T's mind, would be the floating-point arithmetic routines and, above all, the one which prints out numbers in decimal. This last may or may not be in the Basic ROM on any given model, but it would be lovely to get the address(es) and see. Mr T's home-brewed decimal printout routine is very clumsy, and being a linguist rather than a mathematician he can't think of a neat algorithm. If you know of one or know any of the above-mentioned addresses, please send them in.

Anyhow, what you certainly can use in the Beeb are the various MOS routines documented by Acorn, such as OSASCI, OSRDCH and of course that wonderful ragbag of OSBYTEs, on which many Basic functions are based. I went through details of two such OSBYTE calls last month. The full list of MOS calls is in the Master Reference Manual, page D.1-2, and in section N.5 there is a list of 37 Basic commands and the MOS routines which they call.

This is where the writer of a column like this has a problem: how much of the manual does he regurgitate? Not all readers will have these two quite expensive tomes, but for those who do, such rehashing is a waste of time. Mr T's view is that anyone who really wants to write machine-code programs has simply got to find the cash and buy the manual, or Part One at the very least. I can't see how anyone can do without it. What we'll try to do here is to comment on a few of the most useful MOS calls.

What you should be aware of is that calls to the MOS are not 6502 instructions. On a Commodore, JSR &FFEE will not print characters. The difference may be thought to be purely academic, since it will always work on any BBC computer, but let's keep our thinking clear and remember that these are subroutines provided by Acorn, not parts of the 6502 instruction-set.

Secondly, I would always recommend that at the top of your code you declare as variables all the MOS routines which you'll be using, and that those variables be the official names by which Auntie Acorn, God bless her aged bones, baptised them back in the Dark Ages when the Beeb began.

```
100 oswrch = &FFEE
110 osbyte = &FFF4
120 osrdch = &FFE0
```

You will be doing this anyway with parts of your own program, very likely:

```
130 zPageLo = &B0
140 zPageHi = &B1
```

and also with things like vectors which you grab:

```
150 insVecLo = &022A
160 insVecHi = &022B
```

It's so much easier, and you should keep some standard headers on disc for this purpose. But this is one of Mr T's digressions, for which he won the Harold Pinter Memorial Rambling Award in 1991. We'll have a go at a very brief summary of the most useful of these calls:

OSBYTE, &FFF4. The low-level routines which the *FX calls use, plus a few which *FX can't do. You *must* have a book which details these; I can't even begin to list the areas they cover. You can do some things with only the User Guide by putting the *FX number into A and any parameter(s) into X (and Y), but the information in the *FX list is rather limited. Here's an example: *FX21,0 flushes the keyboard buffer. Mr Williams thinks it's too often used; he may be right. *FX21,0 translates to:

```
    LDA #&15:LDX #0 \ no Y parameter
for this one
    JSR osbyte
```

OSWORD, &FFF1, is similar to OSBYTE in that it covers a varied range of functions, but there are only a few calls. You set up a 'parameter block', different for each call, and point XY at it, but you can't do without the layout of the parameter block, which is different for each OSWORD.

The ones you cannot possibly do without are OSWRCH, &FFEE and OSASCI, &FFE3. OSWRCH does precisely what VDU does, (call it with the ASCII code in A), and so does OSASCI with the very important difference that when you call it with A=&0D, it does line feed (CHR$ &0A) *and* carriage-return with the one call. With OSWRCH you have to do both separately. OSWRCH is that teeny bit faster, but it rarely matters. OSNEWL, &FFE7 does a line-feed plus carriage return regardless of the value in A, which can save two bytes. X and Y are irrelevant on entry to all three calls. All registers are preserved on exit, except that, in the case of OSNEWL, A will always hold &0D, but all three calls may corrupt the flags. This is an infuriatingly silly oversight on Acorn's part. In fact, you generally find that the zero flag, which is the one you want preserved in a loop with zero as end-marker, is not corrupted, but Mr T will never risk it.

OSRDCH, &FFEO is like GET$ - it waits indefinitely for a keypress. No entry parameters, X and Y are preserved on exit and A holds the character read. If the carry-flag is set on exit, it normally means that Escape has been pressed, and you should code a response if you don't want your masterpiece to hang up. A BCS to an exit routine is generally indicated.

OSCLI, &FFF7, is the same as Basic's OSCLI - it issues a star command. Point XY at a string in memory, *not* beginning with an asterisk but terminating with &0D, and that string

Continued on page 53

512 Forum

by Robin Burton

Well well, here I am again after all.

Last month we looked at displays up to CGA. This month we'll cover CGA to the present. This won't help you to persuade programs to run on the 512, but you will see why some can't. Also, if you're contemplating a machine upgrade it's an introduction to current PC graphics, which can then be compared with alternatives.

EGA AND BEYOND

CGA was the first PC colour display, but as hardware advanced, so did the demands made on graphics hardware by applications. Soon CGA wasn't enough.

EGA (the Enhanced Graphics Adaptor) came next after CGA. The modes and resolutions supported include all those in last month's table plus the following.

Mode	Resolution
13	320 x 200 16 colour graphics
14	640 x 200 16 colour graphics
15	640 x 350 monochrome graphics
16	640 x 350 4 or 16 colour graphics

EGA DISPLAY MODES

Clearly EGA was quite a leap forward in colours and resolution. At 320x200 the number of colours jumped from four in CGA to sixteen, while 640x200 offered sixteen colours as opposed to CGA's two at that resolution.

Note that, although the maximum horizontal resolution is still 640 pixels (which the BBC can match) vertical resolution increased by 75%, even in colour. It hardly needs saying that 640x350 pixels is way beyond the BBC micro (and the 512). Even without colour 640x350 requires 28K, so if a program demands EGA, forget it as far as the 512 is concerned: everything from now on is for information only.

One point you might notice is that there was no additional DOS text support with the introduction of EGA. Since CGA text surely offers all one could reasonably want in the way of coloured text, it was not surprising that nothing was added. CGA is still the standard for DOS text today, regardless of display type. Note, however that I said 'for DOS'. It's not necessarily so for all PCs.

With the arrival of EGA, extra display RAM could be used for text. Using a specially installed driver, text could be displayed in 25, 43 or 50 lines by 80 or 132 characters, even though these aren't standard in DOS itself.

Interestingly, EGA was also the first standard under which there was some choice over how much RAM to use for the display. The minimum used (in mode sixteen) gave four colours at 640x350 pixels, but simply by adding RAM to your existing card this could be increased to sixteen at relatively low cost.

Calculating display RAM is simple. Four colours require two bits (values 0 to 3) for each pixel, so 640x350 requires 640x350x2 bits for a full screen, or 56K. Sixteen colours require twice the number of bits (values 0 to 15) so 112K are needed.

Notice that this amount of display RAM represents almost twice the total RAM the 6502 processor can address. It's easy to see why no-one ever offered graphics enhancement hardware for the 8-bit BBC - it can't be done. The only way is to completely redesign the machine, then call it something else, such as "Archimedes"!

DOS AND BEYOND

Of course EGA wasn't the end either. Along came VGA (Video Graphics Array) offering higher resolution and more colours again. The range was now from 640x400 in sixteen colours (128K needed) to 640x480 in up to 256 colours (512K). VGA is the minimum you can now buy, but it too has been long surpassed, as we shall see.

VGA was notable as the first display to employ what is now the norm for graphics support in PCs. Bear in mind that, although MS-DOS is up to version 5 and DR-DOS is on version 6 (with 7 soon to be released) MS-DOS 3.3 is still very much current and most PC vendors offer the choice of 3.3 or 5. As you can see from the table, DOS doesn't support VGA, so how's it done?

PC graphics hardware developed faster than DOS, a normal state of affairs when, unlike Acorn, Macintosh, Amiga and others, the hardware manufacturers don't provide the operating system. If you design and supply everything yourself you have to make sure it all matches (more or less). If you don't, as is the case with PCs, it's not your problem.

Display modes already provided by DOS must be maintained for millions of old machine users, otherwise they can't upgrade. Given the huge numbers of users with old display hardware, any other idea is financially out of the question, so existing DOS screen modes are supported by all new versions of DOS. My old XT, for example, runs MDA, CGA and EGA quite happily in DR-DOS 5.

A different approach is needed, one that doesn't rely on, or affect, DOS. The answer, inspired by the PC's original design, was to remove graphics support from DOS. VGA installable graphics drivers began to appear.

Once a method of dealing with a problem has been devised there's often a knock-on effect. Just as EGA was the first display to offer support for text in DOS's built-in modes, VGA was the first to offer support for graphics. This approach is now standard for all current display types, though the original screen modes are still supported by DOS.

CURRENT TRENDS

Over the past couple of years SVGA (Super VGA) and XVGA (eXtended VGA) have become the norm for PC displays. Notice I didn't say 'standard'. Neither of these terms is well enough defined for everyone to agree precisely where the boundaries lie. Suffice it to say, the name doesn't matter, since both exceed VGA resolution by a considerable margin.

Minimum VGA colour graphics stabilised at 600 x 480 pixels in 16 colours, but SVGA or XVGA is what you'll normally get with a new machine these days. If you request a monochrome display it will be monochrome VGA. SVGA offers 800x600 and (some say) 1024x768 resolution, while XVGA offers 1280x1024. Naturally, most SVGA and XVGA boards support lower resolutions too.

It's not easy to specify a standard number of colours for SVGA and XVGA either, since that depends on the amount of RAM fitted and the board type. At their highest resolutions the cheaper (£40.00 or so) SVGA boards only manage 16 colours, but 256 is more usual. SVGA resolution is the minimum if you intend to do much work in Windows or DTP, while XVGA is required for serious CAD.

Naturally, as the resolution of PC graphics increased, so did that of screen hardware, using finer phosphor dots and operating at higher refresh rates. However, be aware that when you get past 800x600 pixels, regardless of colour, a 14-inch monitor really isn't up to the job. This is because, however good a screen's definition or quality, the physical area of a 14-inch screen just isn't enough for the huge amount of data which can be displayed at high graphics resolutions. Large high resolution screens are pretty expensive, but prices are falling as competition increases. SVGA plus a 15 inch screen is a popular choice for home use (constrained by cost) but 17 inch is the minimum for XVGA at 1280x1024.

The number of display colours varies in proportion to the amount of graphics RAM and inversely with resolution. 256 colours at 800 by 600 is the minimum for any self-respecting SVGA card, but the following table shows the options more fully. The first figure is the actual amount of RAM required, while the figure in brackets is the physical amount you'll need on a board to provide it.

Resolution	Colours	Ram needed
640x480	16	153,600 (256K)
640x480	256	307,200 (512K)
800x600	16	240,000 (256K)
800x600	256	480,000 (512K)
1024x768	16	393,216 (512K)
1024x768	256	786,432 (1Mb)
1280x1024	16	655,360 (1Mb)
1280x1024	256	1,310,720 (2Mb)

Display resolution/colours versus RAM

These are the commonest resolutions, but if you really want to push the boat out you can go to 1600x1200 colour graphics too, though screen cost becomes a major consideration at that level. You definitely need a 20 or 21 inch screen and a suitable model will cost (and weigh) more than the average 486 PC.

WHAT NEXT?

Higher resolution with more colours demands more graphics RAM, but another factor becomes significant. Supporting full XVGA in colour can require formidable processing power. Work it out! At 1280x1024 there are over 1.3 million pixels to maintain, while at 1600x1200 the number is almost 2 million (and you've still got to deal with colours).

At 1600x1200 in 256 colours a complete screen re-draw means moving 2Mb of data - after you've processed it all. Clearly this takes time, enough in fact for even today's fastest 486 processors (now 66MHz) to begin to show the strain, with undesirable effects on overall performance. Until very recently the only options were to accept poor performance, or to upgrade to a PC with a faster processor.

Looking ahead a little, it's clearly not possible to continue to push displays much further without a rethink. The quantity of RAM needed and (perhaps more importantly) the power to drive it varies with the product of a display's X,Y resolution. For example a display of 2148x1600 would be almost 3.5 million pixels, which would need 4Mb of RAM in 256 colours.

Display RAM is already independent of the computer's main RAM, while software support is already independent of DOS. The next step is fairly obvious - direct graphics processor support.

The latest (affordable) development in display adaptors are 'graphics accelerator' cards and 'local-bus' main boards. These cards have a dedicated graphics processor (usually an S3) and a high speed, custom colour chip called a RAMDAC, plus typically 2 or 3Mb of RAM. (Let's ignore TIGA boards, they start at over £1000.00!) Local-bus means the main processor has a fast, direct 32-bit data path to the graphics controller, by-passing the slow AT-bus.

Graphics accelerators are rapidly becoming popular, though of course they cost more than normal graphics boards. Typically £300.00 buys a card that, with a suitable screen (hold your breath!) can handle a resolution of 1280x1024 displaying 32,000 colours from a palette of 16.7 million. Some cards support lower resolutions too, if drivers are provided, but at say 640x480 pixels they offer more colours than there are pixels to display them, so some don't!

Accelerator cards relieve the main processor of much of the graphics load, operating quite differently from conventional display methods. The principle is more similar to the page description languages used by laser printers than to a normal display. The main processor tells the graphics card what to draw and where to draw it, but not how to. The graphics processor knows all about geometric shapes, bezier curves, lines, vectors, coordinates and so on, while the RAMDAC supplies the colour. A simplified example of how this works goes as follows.

The main processor tells the graphics processor to display (say) a filled circle of a certain size in a particular shade of a certain colour, centred on screen coordinates of X and Y. This data amounts to only a few dozen bytes. Having told the graphics processor what to do, the main processor can immediately get on with something else while the circle is being drawn by the accelerator card.

DISPLAY DRIVERS

Ever since VGA appeared, graphics and text drivers have taken over display support from DOS. Drivers are installed in the system and loaded as and when when required, bypassing the BIOS often even in text modes. Drivers must be supplied with each graphics card.

Without display drivers most high resolution display cards can produce only CGA text in DOS. In fact some of the more specialised cards can't even do that on their own and a few don't even support DOS text at all. This is one reason, as mentioned last month, for dual screen systems. DOS runs on one (cheap) display, while high definition graphics are displayed exclusively on the other (very much more expensive) one.

Display drivers aren't only specific to a graphics card - they must also match the application or the application's environment. A range of drivers is supplied with graphics cards to cater for a range of the most popular tasks. Most mass-market SVGA and XVGA cards do provide support for standard DOS modes, but nowadays even simple cards employ a different driver for each different graphics resolution.

Drivers for entirely graphical environments such as Windows and OS/2, or for major DOS graphics applications like AutoCAD, are supplied with the majority of display cards. For less popular applications drivers must often be acquired separately. It's usual in such cases for the application itself to provide a range of drivers for the more popular graphics cards.

However (and this might be amusing for 512 users), if you have a specialised graphics card AND you want to use an obscure graphics application, you may well find the two incompatible if no-one has yet written a suitable driver.

THE END? NOT YET!

I hope, after this flight through modern PC graphics you can see why questions about DOS 5 and Windows are frustrating. The subject is complex, and it should be obvious even from our brief look at the subject that such questions are mostly pointless. Not only is there a yawning (and ever-increasing) gulf between the 512's hardware and current PC hardware, the software employed in VGA, SVGA and XVGA display cards is beyond the 80186 processor anyway. It wouldn't work even if the 512 processor was used in a real PC, when we wouldn't be restricted by the venerable old Beeb at the other end of the Tube. It wouldn't even be any help if you could run DOS 3.3 or 5 on the 512, since DOS doesn't handle these new displays.

There is a cheerful note for 512 users though. There are still millions of XTs and even a good many 80286 ATs that can't support more than a megabyte of main RAM. XTs can't run Windows (recommended minimum 4Mb) or OS/2 (8Mb) at all, since they use an earlier processor than the 80186, and they can't run displays more recent than EGA either. 286 machines can run Windows, but they are so slow that they are virtually unusable.

Why is this good news? Well, no-one writes programs for EGA or VGA these days. They either write programs for standard DOS screen modes, or for the latest display standards. Many do both.

This means, leaving aside SVGA and XVGA (and whatever happens next), that the 512 is still reasonably compatible with the majority of PCs in use today, and software is still being written for these machines. Old machines don't make headlines, but there are far too many of them for even the likes of Microsoft to ignore. A recent market projection showed that the number of DOS users is expected to continue to rise for the rest of the decade and that in five years DOS users will still outnumber Windows users four to one.

Speaking of recent technology, you may find this interesting. You probably know that the 80186 was never used as a PC processor, but did you know that it has lately found a new role? It's the processor in at least one type of hard disc caching controller, used to speed up average disc access and data throughput rates. There's life in the old chip yet! B

Machine Code Corner (continued from page 48)

will be issued as a star command. For example: the MOS command *SHOW n lists the text currently assigned to function key *n*, but no call is provided to list all ten keys at one go. The Toad Rom 90 has a star-command *S which does just this. The string EQUS "SHOW 0":EQUB &0D is copied into RAM at &DC00, (which is where star-commands are generally processed by the MOS). A loop is then entered which does:

```
    LDX #0:LDY #&DC:JSR oscli
```

then increments the key-number at the end of the string, round and round ten times. Doing *S on Mr T's Master Turbo thus lists all ten soft-key strings in a neat column - and it's very useful indeed.

There are several other MOS calls, mostly to do with filing, but I reckon the ones given here are the most useful. You really should get the manual, though. David Atherton's excellent "Master Operating System" is also worth the money; Beebug stock both.

This month's competition: will somebody please do some intensive research and tell us exactly when OSWRCH/OSASCI corrupt which flags? Mr T has always intended to find out but never seems to have a spare fortnight. There are still a few 'I'M A SWOT' badges left to be awarded.

That's it for now, frog-fanciers - "And not before time!" I hear you croak. Next month we spill the beans about that legendary gang, FRED, JIM, SHEILA, HAZEL, LYNNE and ANDY, full-colour action photographs and all. We'll also be listing a Tibetan prayer-wheel simulator program which prints OM MANI PADMI HUM every 170 microseconds, fast enough to get even a toad to Nirvana in only 1.2 standard lifetimes. It also lists the Nine Billion Names Of God. B

League Tables (continued from page 15)

```
adminton League."
 3240 PRINT STRING$(38,"=")
 3250 PRINT"Table dated ";d$
 3260 PRINT STRING$(12+LEN(d$),"=")
 3270 FOR T%=1 TO K%
 3280 IF team$(T%,2)<>team$(T%-1,2) PROC
page
 3290 PRINTteam$(T%,1);TAB(17);result(T%
,1);TAB(21);result(T%,4);TAB(25);result(
T%,5);TAB(29);result(T%,2);TAB(35);resul
t(T%,3)
 3300 NEXT
 3310 *FX3,0
 3320 PRINT''TAB(5)CHR$131;"Press SPACE
 to continue.":REPEATUNTIL GET=32:PROCon
:ENDPROC
 3330 :
 3340 DEF PROCpage
 3350 PRINT''"Division - ";team$(T%,2)
 3360 PRINT:PROChead:PRINT
 3370 ENDPROC
 3380 :
 3390 DEF PROCall:T%=1:VDU14:PROCoff
 3400 IF T%>K% VDU15:PROCon:ENDPROC
 3410 CLS:D$=team$(T%,2):PROCdouble(130,
D$,3)
 3420 PRINT':PROCheader:PRINT
 3430 REPEAT:PRINTCHR$134;team$(T%,1);TA
B(18);result(T%,1);TAB(22);result(T%,4);
TAB(26);result(T%,5);TAB(30);result(T%,2
);TAB(36);result(T%,3)
 3440 T%=T%+1:UNTIL team$(T%,2)<>team$(T
%-1,2)
 3450 PRINTTAB(8,22)CHR$131;"Press SPACE
 to continue.":REPEATUNTIL GET=32:GOTO34
00
 3460 :
 3470 DEF PROChead
 3480 PRINTTAB(5)"Team";TAB(16)"Played";
TAB(30)"Rubbers"
 3490 PRINTTAB(21)"W   L";TAB(29)"For
Ag.":ENDPROC
 3500 :
 3510 DEF PROCadd
 3520 CLS:PROCdouble(130,"Add Results",2
)
```

```
 3530 PROCdouble(130,D$,5)
 3540 PRINT'TAB(2)CHR$130;"Teams:"
 3550 FOR T%=1TO K%
 3560 IF team$(T%,2)=D$ PRINTTAB(5);T%;"
. ";team$(T%,1)
 3570 NEXT:PROCon
 3580 PRINT'TAB(2)CHR$131;"Select";:INPU
T" two teams by no.(A,B):"NA,NB
 3590 IF team$(NA,2) <>D$ OR team$(NB,2)
<>D$ VDU7:VDU11:VDU11:GOTO3580
 3600 PRINTTAB(2,18)CHR$134;"Rubbers for
 ";team$(NA,1)
 3610 PRINTTAB(2,20)CHR$134;"Rubbers for
 ";team$(NB,1)
 3620 INPUTTAB(33,18)R1:INPUTTAB(33,20)R
2:PROCoff
 3630 PRINT''TAB(9,22)CHR$131;"Is this c
orrect? (Y/N)":A=GET
 3640 IF A<>89 THEN 3520 ELSE PROCon
 3650 result(NA,1)=result(NA,1)+1:result
(NB,1)=result(NB,1)+1
 3660 result(NA,2)=result(NA,2)+R1:resul
t(NB,2)=result(NB,2)+R2
 3670 result(NA,3)=result(NA,3)+R2:resul
t(NB,3)=result(NB,3)+R1
 3680 IF R1>R2 result(NA,4)=result(NA,4)
+1:result(NB,5)=result(NB,5)+1
 3690 IF R2>R1 result(NB,4)=result(NB,4)
+1:result(NA,5)=result(NA,5)+1
 3700 ENDPROC
 3710 :
 3720 DEF PROCremove
 3730 CLS:PROCdouble(130,"Remove Results
",2)
 3740 PROCdouble(130,D$,5)
 3750 PRINT'TAB(2)CHR$130;"Teams:"
 3760 FOR T%=1TO K%
 3770 IF team$(T%,2)=D$ PRINTTAB(5);T%;"
. ";team$(T%,1)
 3780 NEXT:PROCon
 3790 PRINT'TAB(2)CHR$131;"Select";:INPU
T" two teams by no.(A,B):"NA,NB
 3800 IF team$(NA,2) <>D$ OR team$(NB,2)
<>D$ VDU7:VDU11:VDU11:GOTO3790
 3810 PRINTTAB(2,18)CHR$129;"Rubbers for
 ";team$(NA,1)
 3820 PRINTTAB(2,20)CHR$129;"Rubbers for
 ";team$(NB,1)
 3830 INPUTTAB(33,18)R1:INPUTTAB(33,20)R
2:PROCoff
 3840 PRINT''TAB(9,22)CHR$131;"Is this c
orrect? (Y/N)":A=GET
 3850 IF A<>89 THEN 3720 ELSE PROCon
 3860 result(NA,1)=result(NA,1)-1:result
(NB,1)=result(NB,1)-1
 3870 result(NA,2)=result(NA,2)-R1:resul
t(NB,2)=result(NB,2)-R2
 3880 result(NA,3)=result(NA,3)-R2:resul
t(NB,3)=result(NB,3)-R1
 3890 IF R1>R2 result(NA,4)=result(NA,4)
-1:result(NB,5)=result(NB,5)-1
 3900 IF R2>R1 result(NB,4)=result(NB,4)
-1:result(NA,5)=result(NA,5)-1
 3910 ENDPROC
 3920 :
 3930 DEF PROCzap
 3940 CLS:PROCdouble(130,"Remove Team",2
)
 3950 PROCdouble(130,D$,5)
 3960 PRINT'TAB(2)CHR$130;"Teams:"
 3970 FOR T%=1TO K%
 3980 IF team$(T%,2)=D$ PRINTTAB(5);T%;"
. ";team$(T%,1)
 3990 NEXT:PROCon
 4000 PRINT'TAB(2)CHR$129;"Select";:INPU
T" team to be removed : "R%
 4010 IF team$(R%,2)<>D$ VDU7:GOTO3950
 4020 IF result(R%,1)<>0 PRINT''CHR$129;
team$(R%,1);" STILL HAS RESULTS RECORDED
!":ENDPROC
 4030 PROCoff:PRINT''TAB(9)CHR$131;"Is t
his correct? (Y/N)":A=GET
 4040 IF A<>89 THEN 3940
 4050 FOR Z%=R%TOK%
 4060 FOR Y%=1TO2:team$(Z%,Y%)=team$(Z%+
1,Y%):NEXTY%
 4070 FOR X%=1TO5:result(Z%,X%)=result(Z
%+1,X%):NEXT X%
 4080 NEXT Z%:K%=K%-1
 4090 ENDPROC
```

Troubleshooting Guide (continued from page 29)

programs, known as the CLOSE#0 bug. When a database program is transferring data between memory and disc, it opens what is known as a CHANNEL between the computer and the disc drive. Several channels may be open at any one time, the maximum being five. Each channel is given a number between 17 and 21 inclusive. When the computer finishes amending a file, it closes the channel to that file with the command CLOSE#n, where n is the channel number. There is also a command, CLOSE#0, which *ought* to close all the channels. When this general version is used, however, the channels get closed at their original length rather than their updated length. One solution would be to transfer the database system to ADFS, which does not suffer from this problem - see next month's article.

You may have looked up this paragraph from a reference above, where I have advised curing a fault by typing CLOSE#0. If you are using a Master and think your program has updated files on DFS, you should instead type CLOSE#17 , CLOSE#18 and so on up to channel 21. If the computer gives you a message "Channel", it is telling you that that channel was not open when you tried to close it; ignore this and go on to the next number. By closing the channels individually, you avoid the fault which only affects the general CLOSE#0 command. Note also that there is an operating system command, *CLOSE, which shares the faults of CLOSE#0. More details can be found in BEEBUG Vol.8 No.6 page 42.

Next month, my final article will look at boot files and ADFS problems. Until then, happy discing!

Robol - The Game (Continued from page 8)

```
1110 IF INKEY(-87) G%=TRUE:ENDPROC
1120 IF H%=X% AND V%=Y% ENDPROC
1130 P%=M%?(H%+10*V%):PROCdisplay(H%,V%
,P%-4):M%?(H%+10*V%)=P%-4
1140 PROCdisplay(X%,Y%,4):IF Q%=2 OR Q%
=5 M%?(X%+Y%*10)=6 ELSE M%?(X%+Y%*10)=4
1150 IF Q%=3 AND R%=2 N%=N%-1 ELSE IF Q
%=5 AND R%=0 N%=N%+1
1160 COLOUR3:W%=W%+1:PRINTTAB(6,31)RIGH
T$("0000"+STR$W%,4);TAB(17,31);RIGHT$("0
0"+STR$N%,2);:ENDPROC
1170 :
1200 DEF PROCleft:C%=X%-1+Y%*10:Q%=M%?C
%
1210 IF Q%=0 OR Q%=2 X%=X%-1:ENDPROC
1220 IF X%<2 ENDPROC
1230 D%=X%-2+Y%*10:R%=M%?D%:IF(Q%=3 OR
Q%=5)AND(R%=0 OR R%=2):PROCpush(-1,0)
1240 ENDPROC
1250 :
1300 DEF PROCright:C%=X%+1+Y%*10:Q%=M%?
C%
1310 IF Q%=0 OR Q%=2 X%=X%+1:ENDPROC
1320 IF X%>17 ENDPROC
1330 D%=X%+2+Y%*10:R%=M%?D%:IF(Q%=3 OR
Q%=5)AND(R%=0 OR R%=2):PROCpush(1,0)
1340 ENDPROC
1350 :
1400 DEF PROCup:C%=X%+(Y%-2)*10:Q%=M%?C
%
1410 IF Q%=0 OR Q%=2 Y%=Y%-2:ENDPROC
1420 IF Y%<6 ENDPROC
1430 D%=X%+(Y%-4)*10:R%=M%?D%:IF(Q%=3 O
R Q%=5)AND(R%=0 OR R%=2) PROCpush(0,-2)
1440 ENDPROC
1450 :
1500 DEF PROCdown:C%=X%+(Y%+2)*10:Q%=M%
?C%
1510 IF Q%=0 OR Q%=2 Y%=Y%+2:ENDPROC
1520 IF Y%>26 ENDPROC
1530 D%=X%+(Y%+4)*10:R%=M%?D%:IF(Q%=3 O
R Q%=5)AND(R%=0 OR R%=2) PROCpush(0,2)
1540 ENDPROC
1550 :
1600 DEF PROClevel:*FX15,1
1610 REPEAT:CLS:VDU 19,0,4;0;:COLOUR2:I
NPUTTAB(2,10)"Enter level "L%:UNTIL L%>0
AND L%<11:L%=L%-1:G%=FALSE:ENDPROC
1620 :
1700 DEF PROCscreen
1710 VDU 20:B%=L%*160+&3000:K%=0
1720 FOR I%=0 TO 159:S%=B%?I%:M%?K%=S%
DIV 15
1730 M%?(K%+1)=S% AND 15:K%=K%+2:NEXT
1740 K%=0:FOR I%=0 TO 30 STEP2:FOR J%=0
TO 19
1750 S%=M%?K%:PROCdisplay(J%,I%,S%):K%=
K%+1
1760 IF S%=4 X%=J%:Y%=I%
1770 IF S%=2 N%=N%+1
1780 NEXT,
1790 PRINTTAB(0,0);SPC20;TAB(0,31);SPC1
9;CHR$30:COLOUR1:PRINTTAB(0,0)"Lev:";:CO
LOUR3:PRINT;RIGHT$("00"+STR$(L%+1),2);:C
OLOUR2:PRINT" ROBOL  ";:COLOUR3:PRINT"00
:00";
1800 COLOUR1:PRINTTAB(0,31)"Moves:";:CO
LOUR3:PRINT;"0000";:COLOUR1:PRINT;" Pack
s:";:COLOUR3:PRINT;RIGHT$("00"+STR$N%,2)
;:GCOL0,0:MOVE0,32:DRAW1279,32:GCOL0,3:E
NDPROC
1810 :
1900 DEF PROCdisplay(X%,Y%,A%)
1910 CALL &B00:ENDPROC
1920 :
2000 DEF PROCpush(I%,J%)
2010 PROCdisplay(X%+I%,Y%+J%,0):IF R%=2
PROCdisplay(X%+2*I%,Y%+2*J%,5):M%?D%=5
ELSE PROCdisplay(X%+2*I%,Y%+2*J%,3):M%?D
%=3
2020 X%=X%+I%:Y%=Y%+J%:SOUND 1,-10,20,1
:ENDPROC
```

```
 10 REM Program RobData
 20 REM Version B 2.2
 30 REM Author  M.Bobrowski
 40 REM BEEBUG  April 1993
 50 REM Program subject to copyright
 60 :
100 MODE 7:A%=&5400:HIMEM=A%
110 FORI%=A%TOA%+&27FC STEP4:!I%=0:NEX
T
120 FORI%=0TO&63F STEP4:READ A$:I%!A%=
EVAL("&"+A$):NEXT
130 err%=FALSE:FORN%=1TO10:READ checks
```

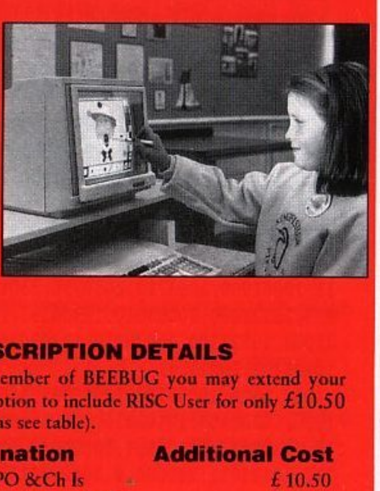
```
um%
  140 S%=0:FORI%=0TO&9F:S%=S%+I%?A%:NEXT
  150 IF S%<>checksum% PRINT''"Error in D
ATA for Screen ";N%:err%=TRUE
  160 A%=A%+&A0:NEXT:IF err% VDU7:END
  170 OSCLI"SAVE robol3 5400 +1080 3000
3000"
  180 END
  190 :
  200 REM Scr 1
 1000 DATA 0,0,0,0,0,0,0,0,111101,0,1000
0,1,0,13001,0,110100,1031,1000000,103030
00,0
 1010 DATA 1011101,11011010,11011,111001
01,10220011,30000301,0,11011022,1110111,
10220041,10000,11110100,1011,11111101,0,
0,0,0,0,0
 1020 REM Scr 2
 1030 DATA 0,0,0,0,0,0,0,1000000,1111111
1,1011,220100,11000010,1000010,30100022,
100003,220100,111113,1000010,40000022,10
0011
 1040 DATA 220100,1031010,1000010,111011
11,103030,3010000,30303000,10,10000001,1
00000,11010000,11111111,10,0,0,0,0,0,0,0
 1050 REM Scr 3
 1060 DATA 0,0,0,0,0,0,0,0,11010000,1011
11,0,4000001,10,3010000,100113,0,3100030
1,0,13010000,103
 1070 DATA 11111100,1010311,12000011,301
2022,13000,20221100,33000,12000001,11012
022,111111,11111100,11,0,0,0,0,0,0,0,0
 1080 REM Scr 4
 1090 DATA 0,0,0,1000000,10111111,0,2200
0100,11111022,11111111,10222200,10010,22
003030,33101022,1033031,10222200,310,220
00103,33101022,31303001,10111111
 1100 DATA 10310,100,1110000,11111111,0,
10010,1000,100000,11003000,0,33310310,41
00,100000,11000001,0,11111111,1011,0,0,0
 1110 REM Scr 5
 1120 DATA 0,0,0,11010000,11,0,11010001,
11,1010000,11031,0,3000001,11000001,1111
111,10011,20221200,31000301,12000011,300
02022,100133
 1130 DATA 20221200,4033001,11000010,111
111,100130,0,30301,10,11010000,101001,0,
100,10,1000000,101111,0,0,0,0,0
 1140 REM Scr 6
 1150 DATA 0,0,0,0,0,11110000,10110011,0
,1012012,1114,20120000,1001101,0,2012,13
3,20120000,1030101,0,1112112,103
 1160 DATA 11110000,1300103,0,31000100,1
03,1000000,1300003,0,11000100,100,100000
0,11111111,0,0,0,0,0,0,0,0
 1170 REM Scr 7
 1180 DATA 0,0,0,0,0,0,10111100,0,101111
11,1100,10100100,1331041,1000000,300000,
100,30000100,1001101,1000000,11110111,11
31
 1190 DATA 30100,10221011,1000000,203030
3,1022,100,10221211,1000000,12103003,102
2,11000100,10111110,1000000,1011,0,0,0,0
,0,0
 1200 REM Scr 8
 1210 DATA 11110000,0,0,11110110,111111,
100000,3030003,1,1033110,13000,3100000,1
00003,11000001,13110,11111,3131400,11000
3,10000001,31010300,10100
 1220 DATA 30001000,3030300,11000001,110
11011,111111,100000,100,0,1000010,0,2212
0000,2122,0,21222212,0,22120000,2122,0,1
1111111,0
 1230 REM Scr 9
 1240 DATA 0,0,0,1000000,111111,0,220001
00,21,11110100,212200,1000000,20000000,2
1,11000100,212200,1000000,22001110,21,11
101100,111111
 1250 DATA 10000000,113033,11000000,3031
011,111101,13001001,10003,40010001,330,1
0330,10111101,11010333,11,1000,1,1100000
0,111111,0,0,0
 1260 REM Scr 10
 1270 DATA 11001101,11111111,41111011,10
11,10001000,3003310,2030330,3101022,3001
033,10221200,10003010,12300333,10111022,
31000,10221200,10000010,12303030,101022,
10111101,10221211
 1280 DATA 10000111,12000303,1101022,333
1010,10221103,10102210,3,22101012,303310
10,10123033,10101111,10000000,1012,11111
110,10121011,100000,0,1012,11111111,1011
1111
 2510 REM Checksum data
 2520 DATA 881,1106,789,1715,1016,755,10
26,1223,1232,2531
```

New Generation!

RISC *user*

RISC User, the highly popular magazine for Archimedes users, is bigger and better. The new RISC User is now B5 size which offers a sophisticated design, bigger colour illustrations and bigger pages with more information. Altogether better value and no increase in price.

RISC User is still a convenient size to assemble into an easy-to-use reference library, containing all the information you need as an Archimedes user. Every issue of RISC User offers 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. Altogether RISC User has established a reputation for accurate, objective and informed articles of real practical use to all users of Acorn's range of RISC computers.



YOUR ARCHIMEDES ON THE PHONE

A guide to communications introducing the readers to bulletin boards, file transfer protocols, conferencing and comms software.

REVELATION IMAGE PRO

A review of this impressive painting package from Longman Logotron.

RISC OS 3 SCREEN MODES

A comprehensive look into the range of Desktop modes available to Arc users.

THE PC EMULATOR SURVIVAL GUIDE (1)

New series on how to use the PC emulator which starts with introducing the PC world.

FAXPACK UPDATE

A look at the latest addition to Computer Concept's Fax Pack which now allows you to use your computer as an answering machine.

QUICK INDEX

A useful index generator for books and manuals.

WRITE-BACK

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

WP/DTP

Articles on using different DTP and WP packages.

INTO THE ARC

A regular series for beginners.

TECHNICAL QUERIES

A column which answers your technical queries.

NOOT

Review of the new animation package for the educational market.

SUBSCRIPTION DETAILS

As a member of BEEBUG you may extend your subscription to include RISC User for only £10.50 (overseas see table).

Destination	Additional Cost
UK,BFPO &Ch Is	£ 10.50
Rest of Europe and Eire	£ 15.40
Middle East	£ 19.60
Americas and Africa	£ 21.90
Elsewhere	£ 33.00

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

Please do keep sending in your hints for all BBC and Master computers. Don't forget, if your hint gets published, there's a financial reward.

BIT FIELDS IN BASIC

Nick Mellor

Following last month's tip for extracting a single bit from a value returned by USR, here are some further ideas on a similar theme: squeezing several small pieces of information (fields) into a single byte.

The first function, *store_field*, is used to store a small number within a byte between specified bit positions.

```
   DEFFNstore_field(byte%,start_bit%, stop_
bit%,data%)
   LOCAL bits%,mask%,a%
   mask%=0:bits%=stop_bit%-start_bit%+1
   bits%=stop_bit%-start_bit%+1
   FOR a%=1 TO bits%:mask%=mask%*2+1:NEXT
   data%=data%ANDmask%
   bits%=7-stop_bit%
   IF bits%<>0 THEN FORa%=1TObits%: data%=d
ata%*2:mask%=mask%*2:NEXT
   =(byte% AND NOTmask%) EOR data%
```

FNstore_field is called with four parameters. The first is the original byte of data in which the field is to be inserted. The second and third parameters are the start and stop bit positions, numbered from 0 to 7 and from left to right. The fourth is the field value to be inserted into the original byte. The byte returned will have the new field value inserted.

The next function can be used to extract fields from a byte. The function takes three arguments: the first is the byte to read from, and the second and third are the start and stop bit positions, numbered as for *FNstore_field*.

```
   DEF FNfield(byte%,start_bit%,stop_bit%)
   LOCAL mask%,bits%
   bits% = stop_bit%-start_bit%
   mask%=&FF DIV (2^(7-bits%))
   =(byte% DIV (2^(7-stop_bit%)) AND mask%)
```

The functions can be used as a pair to save memory by storing several small objects in one byte. Doing so can make a huge difference to the amount of memory used as against storing true-false flags or small integer ranges in integer variables or even bytes. For example, storing a number whose range is from 0 and 7 (3 bits wide) could be achieved by:

```
?loc%=FNstore_bits(?loc%,5,7,field%)
```

and can be read using:

```
field=FNfield(?loc%,5,7)
```

Such sub-byte manipulations are the basis of many compression algorithms.

INSTANT ITALICS

Al Harwood

Use the following procedure in any program for instant italic text in modes 0 to 6:

```
   DEFPROCitalic(text$)
   LOCAL A,A%,A$,X%,Y%
   FOR A=1 TO LENtext$:A$=MID$(text$,A,1)
   ?&70=ASCA$:X%=&70:Y%=0
   A%=10:CALL&FFF1
   !&79=&6A0070B9:!&7D=60007099
   FORY%=1 TO 8:IF Y%=4 ?&7C=&2A:Y%=6
   CALL&79:NEXT
   VDU23,128,?&71,?&72,?&73,?&74,?&75,
   ?&76,?&77,?&78,128
   NEXT:ENDPROC
```

The routine uses locations &70 to &80, and redefines the character with code 128. The first character of text$ is printed in italics at the current cursor position.

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.

BBC B, Opus DDOS, dual D/S 40/80 & single D/S 40/80 disc drives with PSU's, Aries ROM/RAM board with shadow RAM and many ROMs, 6502 2nd processor, Philips 12" mono monitor, all manuals, all BEEBUG issues to date and Acorn User issues 2-65 inc. all in binders, software with manuals, text books plus other extras. Tel. 081-551 5648 days/eves.

WANTED: Copy of Viewsheet/Viewstore & 5.25" utility disc - by Dabs Press. **WANTED:** ATPL sideways ROM board for BBC B+. Tel. (0734) 345959.

3xM128's with various ROM software £150 each, 2xMicrovitec Cub RGB £50 each, 1x Taxan RGB monitor £75 please ring for details on the above Masters all in excellent condition. Tel. (0453) 885139.

A310 package plus extras £700 o.n.o. Tel. (0788) 832267.

WANTED: Computer Concepts' Spellmaster ROM complete with documentation and manual for BBC Master, must be in perfect working order. Tel. (0955) 81 243.

Micro User magazines complete set from Vol.1 No.1 to Vol.10 No.8 many in binders complete with cassettes Vol.1 No.1 to Vol.3 No.1 and 80T DFS discs from April '89 to Oct '92 £60 o.n.o. Also BBC Prestel Adaptor and ROM still in original box £30 o.n.o. Buyer (s) collect. Tel. (0742) 468969.

WANTED: Could anyone please lend or sell me a manual for Intersheet? Tel. (0821) 642652 eves.

WANTED: BBC B or Master in reasonable condition for my handicapped son - must be cheap. Tel. (0821) 642652 eves.

WE Video Digitiser £60, Spellmaster, PMS Multifont NTQ + utility & 4 font discs, PMS Publisher + utility & 3 fonts discs, Stop Press + extras 1&2 7 fonts/graphics discs - all £25 each, ADI, ADT, Snatch, Interword, Music Processor V2, CJE Multifont NLQ + 4 fonts discs, Graphito, Navex v3.1 + charts, Holed Out Extra Courses 1&2, Nidd Valley Digimouse - all £10 each, MOS+, Dumpout 3, Scythe (inc. utility disc), Procyon, Pen Friend, Dumpmaster, Helping Hand, Adv printer buffer, Vector 2, Colossus Bridge, Voltmace Delta 3b joystick - all £5 each. Tel. (0883)345294 eves.

AMX SuperArt £10, System Delta £10, Peartree Business System £10, Edword2 wordprocessor £5, Dumpmaster £6, Toolkit £4, Exmon £4, Viewstore £15, Printer Driver (View) £3, Artist (Peartree) £8, Discmaster £2, GrandPrix £4, Brother IF-50 typewriter interface £15, all originals with handbooks etc. Tel. (0623) 27423.

BBC B issue 3 with DDFS, 32k Shadow RAM and Sideways ROM/RAM boards, View Professional, Toolkit Plus, Exmon, Sleuth, 5.25" & 3.5" DS 80T disc drives, Revs, Aviator £160, BEEBUG magazines Vols. 1-11 complete and bound £60. Tel. (0923) 239788.

Stars and Planets plotted for any date and time, BBC Master or equivalent, 80T on 5.25" disc, nominal charge for disc and postage. Tel. (0932) 873278.

WANTED: Copy of Dabs Press Hyperdriver for View either on ROM or disc for sideways RAM, complete with documentation, secondly a spellchecker for View preferably on ROM but disc would do. Tel. (0274) 505288.

Free to charity, cheap to a good home BBC B and Master 128 software, phone for details. Taxan Kaga KP810 9 pin DM printer inc. RAM chip, vgc only £65. BBC Master 128 ROM cartridge with Exmon II and WE NLQ Designer and many NLQ fonts on disc to download to KP810 RAM £30 (both items £80). Tel. (0283) 31403 anytime.

WANTED: Monitor for BBC B preferably Microvitec 14" model 1451 or 1441. Offers (0602) 654426 after 6pm.

Master 512 in excellent working order, included are two 5.25 floppies, AMX mouse, joysticks and the following EPROMs in addition to the resident View and Viewsheet: Epson printer driver, Wordwise Plus, Pascal, Screendump, Toolkit and Graphics, full original manuals and discs for the Master 512 plus various extra manuals, years of BBC dedicated magazines, software for both BBC & PC mode including games. Only £325 o.n.o. Will deliver north of England. Tel. (0535) 662157.

Music software; Music Master with microphone interface, 5.25" disc, handbook.Mupados Recorder Tutor with Ensemble, Duet and Classroom packs (5x5.25" discs, handbooks and cassettes), Micro Maestro with 5.25" disc and 6 cassettes, all for £32 including postage (worth £179). Tel. (0256) 27018.

WANTED: Teletext adaptor. Tel. 081-539 7607 eves.

WANTED: Snooker game, also manual for BEEBUG Wordease, buy or copy, Cumana 40/80T dual drive. Tel. (0637) 873788.

WANTED: 256k printer buffer made by Watford Electronics, may consider a faulty unit, if cheap. Tel. (0294) 52250 eves.

WANTED: Dumpmaster II ROM with instruction book for my BBC Master 128. Tel. (0751) 73342.

WANTED: Combined 5.25" and 3.5" drive for Master, manuals for Ovation (mine lost) and Viewsheet, copies of Peter Killworth's "How to Write Adventure Games" and "Creative Assembler on the BBC Micro". Tel. (0279) 813463 after 6pm.

Loads of BBC B games and some utility programs, you will need a double sided 80T disc drive to operate them £50 o.n.o. Tel. (0707) 655132 after 6pm.

Tandata Td1400 Viewdata terminal, 1200/75bps, little used, perfect condition with manual. Offers? Tel. (0928) 722454.

A5000 4Mb RAM, Acorn multisync, Learning Curve with PC Emulator v1.8, various software, discs, books etc. £1400, Acorn Desktop C £150, Clares Illusionist £50, CC Compression £25, RISC User complete £30, Archive complete £30, BEEBUG complete £30, v21/22 /23/22bis/42/42bis MNP 4/5 modem, terminals plus, Arc to Hayes lead £150. Or all for £1700. Tel. 081-698 3772.

WANTED: BBC hard drive 20 or 40Mb, any make, PC software CAD, Deluxepaint II, GEM 3, Shibumi Problem Solver 5" disc, good PC wordprocessor, PMS real time Genie clock, also Pace eurolink modem - autodial/answer. **FOR SALE:** SoundCAD £25, Superdump £25, Dumpout 3 £7, Wapping Editor £25, GXR B+ ROM £10, Termulator B+ ROM £10, Office Mate £5, Office Master £5, 12 cassette games £10 the lot. Also 12 BBC books. Write with s.a.e to; Mr C Game, 24 Grosvenor Close, Tiptree, Colchester, Essex CO5 0JN.

EPROM blower with software in EPROM £15, Master Smart cartridge £15, Master ROM £20, Overview suite £40, Contex Bank Manager with Business utilities (Master version) £20, Gemini money management (BBC) £5, Play it Again Sam 13, Repton 3, Repton 4, Citadel, Barbarian, Sim City all £8, Death Star £5. Tel. (0263) 78488.

RINGING THE CHANGES

In thinking of possible ways to solve the BEEBUG Christmas competition my first ideas revolved around letting the computer do most of the work. After finding the three most obvious constants, it would be a matter of testing permutations for the remaining unknowns. The trouble was to find an algorithm to produce the rows; my change-ringing theory might possibly have done the job, but it was going to be a cumbersome business writing a suitable program. I gave in and used brute force (and not a little ignorance).

In all the articles I have read on maths topics, I don't remember seeing anything on producing permutations and wonder if any other readers have any methods of doing so.

R.J.Lindsell

PROBLEMS OF AN AGING BEEB

I find I'm starting to get curious things happening to my BBC model B (purchased October 1982). Please don't suggest that it's most likely to be the power supply. This, with respect is the standard answer. I'm still on my original power supply, but had I taken previous advice I would now be on my 4th or 5th.

My Beeb is 10 years old. A few years ago I converted it to ADFS and replaced the disc controller chip with the 1770 version. Over the years I have physically removed the power supply away from the computer (to assist in cool running), and this is now attached by an umbilical cord. I've also cut a hole in the Beeb's case and fitted a fan, again to assist in cooling. I use the Beeb most days, and it is usually switched on for 10 hours at a time.

Recently I've experienced the Beeb simply switching itself off: in other words, all power has vanished as if it had been switched off or blown a fuse. The monitor, disc drive etc. remain powered up. Switching off, preparatory to checking fuses etc., but just switching back on again first, and all worked and remained working for another month or two.

Question: what is failing, and is it something I can easily repair/replace myself? I can't afford to be without the use of the Beeb, and I can't afford to buy an A3000 or any other machine. Any useful suggestions will be gratefully received.

Ian Crawford

No electronic equipment can be expected to continue working indefinitely, but as time goes by spare components are likely to become more and more difficult to find. Despite Mr.Crawford's initial plea, an old power supply can be the cause of problems such as described, particularly if total failure is encountered. We would suggest checking carefully, with the machine unplugged and switched off, that all the connections to and from the power supply are examined for any weak or suspect joints.

Power supplies can eventually fail, and given the decreasing availability of spares, it could be a prudent move to purchase a spare power supply and store this for when that eventuality arises, assuming that there is a definite intention of retaining your Beeb for as long as possible.

On the other hand, machines such as the new A3010, and secondhand A3000s (see back cover) are available at reasonable prices, and it may still be possible to obtain some resale value for your BBC micro. This is an issue which we will be looking at in future issues of BEEBUG.

BEEBUG MEMBERSHIP

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

BEEBUG SUBSCRIPTION RATES

BEEBUG		BEEBUG & RISC USER
£18.40	1 year UK, BFPO, Ch.I	£28.90
£27.50	Rest of Europe & Eire	£42.90
£33.50	Middle East	£53.10
£36.50	Americas & Africa	£58.40
£39.50	Elsewhere	£62.50

BACK ISSUE PRICES

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

Volume	Magazine	5"Disc	3.5"Disc
6	£1.00	£3.00	£3.00
7	£1.10	£3.50	£3.50
8	£1.30	£4.00	£4.00
9	£1.60	£4.00	£4.00
10	£1.60	£4.75	£4.75
11	£1.90	£4.75	£4.75

POST AND 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.

Stock 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

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 840303, FAX: (0727) 860263
Manned Mon-Fri 9am-5pm (for orders only 9am-6pm and 9am-5pm Saturdays)
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by RISC Developments Ltd.

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

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

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

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

In all communication, please quote your membership number.

RISC Developments Ltd (c) 1993

ISSN - 0263 - 7561

Printed by Arlon Printers (0923) 268328

Magazine Disc

April 1993

ADFS BACKUP - the second part of Roger Smith's ADFS backup utility. This is a front-end program for the Fast Backup utility presented in the last issue.

ADFS DISC COMPARISON - Roger Smith's utility for comparing two ADFS discs.

DATASHEET 2 - an upgrade for Stephen Colebourne's spreadsheet "DataSheet" that appeared in Vol. 11 Nos. 1 & 2.

FRACTIONS - utility to print fractions in Interword and/or Wordwise.

LEAGUE TABLES - an indispensable aid to the badminton league secretary! Can easily be adapted for other purposes.

WORD SQUARES - an upgrade to Graham Leng's word search program presented in Vol 11 No. 7.

PLANET AND MOON ORBIT - an interesting simulation of a space-ship orbiting the Earth and Moon (circular and figure of eight orbits). Orbital motion can be viewed as it evolves naturally, or you can perturb the orbit slightly and watch the effect.

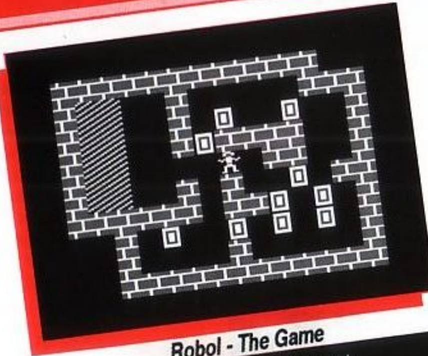
DEMONSTRATION OF INKEY(n) and INKEY(-n) - Example program from 1st course by Alan Wrigley.

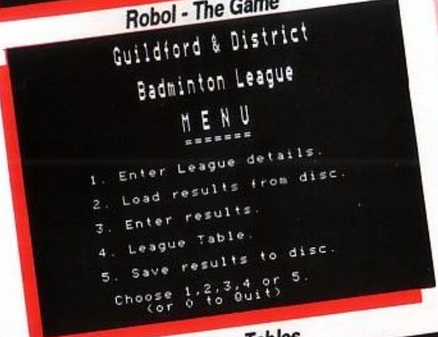
ROBOL - a BBC version of the strategy game Sokoban for the IBM PC.

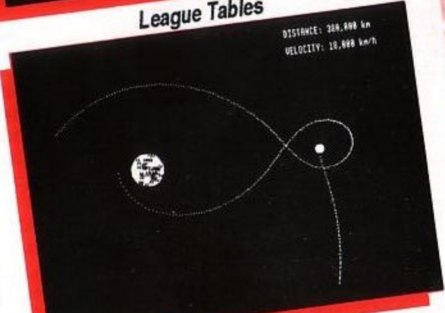
SPRITES - examples from Alan Blundell's examination of Acorn's undocumented graphics facilities for the Master.

SORTING - example program that demonstrates the Bubble and Shell sorting methods dynamically.

MAGSCAN DATA - Bibliography for this issue (Vol. 11 No. 10)


Robol - The Game


```
Guildford & District
Badminton League
MENU
=======
1. Enter League details.
2. Load results from disc.
3. Enter results.
4. League Table.
5. Save results to disc.
Choose 1,2,3,4 or 5.
(or 0 to Quit)
```
League Tables



Gravity and Orbits

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.

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

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

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

Upgrading to an Archimedes

We know that many BEEBUG readers have already upgraded to an Archimedes, and no doubt many more will choose to follow a similar route. For their benefit we offer our advice to help them make a sensible decision on whether to upgrade and if so, what path to take.

Any prices quoted relate to our associated company Beebug Ltd., but note that all prices, particularly those on trade-ins and secondhand items, are likely to change without notice. You should always telephone or write for the latest information.

Archimedes A5000

What System to Choose

All new Archimedes systems are now supplied with the RISC OS 3.10 operating system. Any secondhand system should be upgraded to this. Based on the experience of existing users, we would strongly recommend a minimum of 2Mb of RAM. Most users find a hard disc adds significantly to the convenience of using an Archimedes, but you can always add a low-cost hard drive later, and more memory, but check on the likely price of future expansions - it is not necessarily the same for all machines. If you might be interested in more specialised add-ons (scanners, digitisers, etc.) then check the expansion capability of your preferred system.

Compatibility and Transferability

You will need to decide to what extent you wish to continue using existing discs and disc drives on an Archimedes. An Archimedes and a BBC micro can be directly connected for transfer of files. You can also connect a 5.25" drive to an Archimedes via an additional interface to continue to access 5.25" discs (ADFS format). Our DFS Reader will also allow files to be transferred to the Arc from DFS format discs. However, none of this is possible with the latest A3010/A3020/A4000 systems.

Much BBC micro software will run directly on an Archimedes, or via the 6502 emulator. However, consider this carefully; in our experience, despite prior misgivings, most Archimedes users find that they rapidly adjust to the Desktop environment of the Archimedes, and quickly abandon the software and data of their old system after an initial period.

Software for the Archimedes

The Archimedes is supplied complete with a range of basic applications software. Before embarking on any further purchases it may be better to familiarise yourself with the new machine. Most users look for a word processor (or DTP package), maybe a spreadsheet, or a database, plus other more specialist software. We cannot give detailed guidance here, but back issues of RISC User contain a wealth of useful information - we can advise on suitable issues.

General Advice

It is advisable to discuss your requirements with the BEEBUG technical team before making a final decision on what you want. Try to anticipate future expansion needs at the outset. Note: the price on some systems includes a monitor; in other cases a choice of monitor is available at an additional cost. The details given in the table are minimum specifications of the different Archimedes models.

The A3010

It may also be possible to trade in an existing monitor and/or disc drive, but check if your existing monitor is suitable for use with an Archimedes first. You may find it better to advertise your BBC system in BEEBUG and sell privately - this applies particularly to any software and hardware add-ons which cannot be accepted for a trade-in. In future, all personal ads for Archimedes systems in RISC User will also be included in BEEBUG. You may also defer a trade-in until a later date provided you make this clear at the time of purchase.

Archimedes Systems - Typical or Current Prices

		Secondhand	New
+	A310 1Mb RAM	£350	
+	A410/1 1Mb RAM	£565	
+	A420/1 2Mb RAM, 20Mb hard drive	£650	
+	A440/1 4Mb RAM, 40Mb hard drive	£725	
+*	A3000 1Mb RAM	£350	
*	A3010 1Mb RAM, Family Solution		£ 499.00
*	A3020 2Mb RAM, 60Mb hard drive		£1056.33
*	A4000 2Mb RAM, 80Mb hard drive		£1115.08
*	A5000 2Mb RAM, 80Mb hard drive		£1643.83
+*	Acorn standard colour monitor	£145	£ 258.50

All systems above include a single floppy disc drive.
New (*) and secondhand (+) - all prices inc. VAT.
The A5000 price includes a multiscan colour monitor,
A3020/A4000 price includes standard colour monitor.

BBC Micros - Typical Trade-in Prices

Model B (Issue 7)	£ 35
Model B (issue 7) + DFS	£ 75
Master 128	£125
Master Compact	£ 50

For further information on all Archimedes systems contact:
BEEBUG Ltd. 117 Hatfield Road, St Albans, Herts AL1 4JS. Tel. 0727 840303 Fax 0727 860263