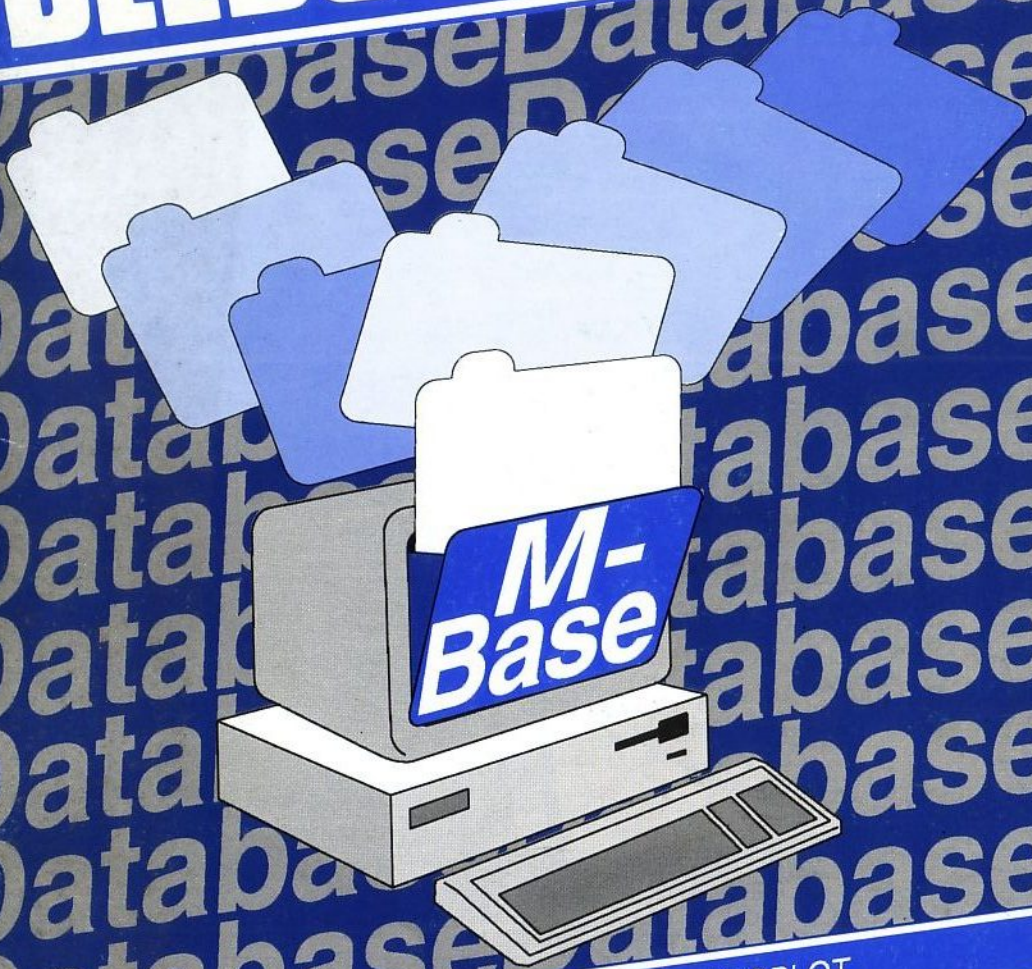
Vol.12 No.6 November 1993

BEEBUG

FOR THE BBC MICRO & MASTER SERIES



- MISSILE NAVIGATION
- SPIROPLOT
- FINANCIAL FUTURES
- MANIC MECHANIC

BEEBUG Vol.12 No.6 November 1993

FEATURES

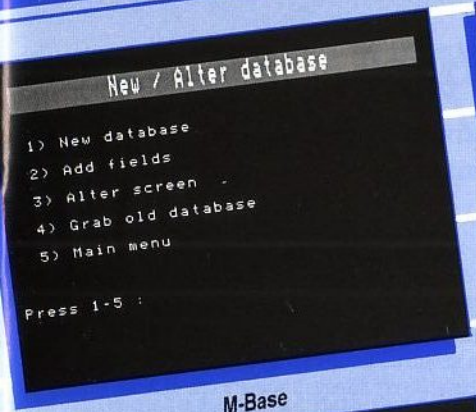
REGULAR ITEMS

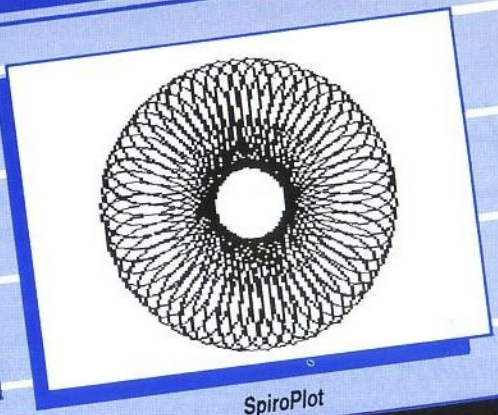
HINTS & TIPS

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 |.

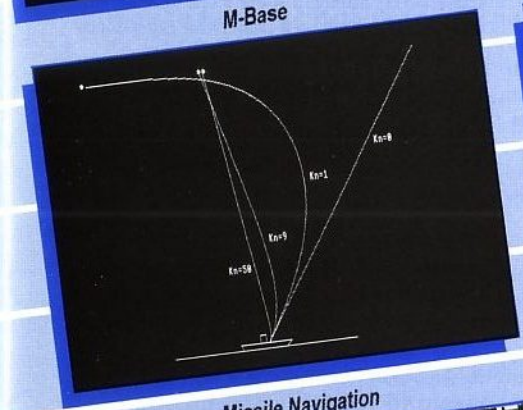
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints





SpiroPlot

M-Base



Missile Navigation

Financial Futures

Manic Mechanic

"... the role of the teacher is central and the needs for inservice and opportunities to experience or learn about new approaches and reflect on practice are paramount"

BEEBUG Education

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

AUTUMNAL REFLECTIONS ON THE BBC MICRO

The onset of autumn traditionally marks a resurgence of interest by many computer users as the nights draw in. Indeed, I expect the clocks will have been put back by the time you read this, so what better than to curl up in front of your computer.

You will see that this month's *Postbag* page is devoted to one topic, and that is a review which we published in the previous issue of a product called Wordwise-Mail. Now I find this correspondence interesting for a number of reasons.

There are very few new products released for the BBC micro these days, and those that are often emanate from single individuals who cannot afford much in the way of advertising. We are only too happy to consider reviewing such products, but we need the information about the product in the first place, and we need someone to volunteer to do a review (for example, I have for review some recent products relating to Inter-Word and Inter-Base - see *News* in BEEBUG Vol.12 No.4).

It is also revealing to see, that of the three main word processors for the BBC micro (Wordwise, Inter-Word and View), by far the greatest interest seems to revolve still around Wordwise (and Wordwise Plus) - see, for example, the latest *Wordwise User's Notebook* in this issue. Yet Wordwise is certainly the oldest of the three. If any readers can fathom out why that is then let us have your thoughts on the subject.

What this all amounts to, as I have said before, is that there is clearly a dedicated band of Beeb users still very actively using their BBC micros for a variety of purposes. It would be nice if we could reveal more of these activities to other readers. This month's letters show that the interest is there and is still very strong. If you are one of those still active users, then why not share your experiences with other readers?

Mike Williams

ALL FORMATS COMPUTER FAIRS

All Formats Computer Fairs are flourishing throughout the country, and details of events for the remainder of this year are given below:

Nov 6	Cowley Parish Hall, Oxford.
Nov 7	Corn Exchange, Church St, Brighton.
Nov 13	National Motorcycle Museum, NEC, Birmingham (J6 M42).
Nov 14	University Union, Park Place, Cardiff. Nov 20 Sandown Park, Esher, Surrey (J9/10 M25).
Nov 21	Guildhall, Portsmouth.
Nov 27	Haydock Park Racecourse (J23 M6).
Nov 28	Bristol Exhibition Centre, Nr. Watershed.
Dec 4	De Montford Hall, Granville Road, Leicester.
Dec 5	Washington Leisure Centre, Dist. 1.
Dec 11	National Motorcycle Museum, NEC, Birmingham (J6 M42).
Dec 12	University Sports Centre, Calverley St, Leeds.
Dec 18	Adam House, Chambers St, Edinburgh.
Dec 19	City Hall, Candleriggs, Glasgow.

Admission costs £4 for adults (£2 after 2pm), £2 for children, wheelchair users free. For 50 £1 off vouchers send a stamped addressed envelope to Bruce Everiss, Maple Leaf, Stretton-on-Fosse, Moreton-in-Marsh, Gloucestershire GL56 9QX, or telephone 0608 662212.

M-Base (Part 1)

I.J.Palmer presents the first part of a comprehensive database for the Beeb.

M-Base is a data handling program which is versatile whilst being easy to use. It has all the features of a simple 'card box' program but with powerful features to allow a more complex database to be managed.

M-Base has four different searches, including a powerful 'closeness' search. Also, there is automatic sorting which means that you do not need waste time sorting data once you have typed it in. There is also automatic housekeeping which means that when you delete records the data is shuffled to reduce the overall size.

Perhaps the most useful feature of M-Base is the way it handles memory. With most programs of this kind you either load the data into memory while you look through it, alter it, etc. or you keep the data on disc all the time. The former of means fast access to data but limits the size of the database you can use. The latter allows much larger databases (up to the size allowed by the filing system) but is often slow, particularly when searching.

M-Base allows you to have the best of both worlds. While the database will fit in memory you can load it into memory when you use it, but once it gets too large for your machine's memory you can then resort to disc use without any changes necessary. When the memory option is used the data is loaded into sideways RAM, allowing just under 63K of data in memory on the Master 128 or Compact.

As the program stands it will only work on Master computers as it assumes four banks of sideways RAM accessed via the SR commands (e.g. SRDATA, SRREAD and SRWRITE) although the program can easily be altered to allow its use on other computers by taking out these commands and using it in disc mode only. You should also note that the program uses the ON...PROC command and OSCLI, these will only work from Basic 4 and 2 respectively and will need to be written in other ways on earlier versions.

This program has been split up into three parts due to its size, although these will be combined to produce a single program. You should take care when typing it in that you notice the line numbering of the final ten lines in the listing included with this section.

Once you have typed in the first part of the program and saved it you are ready to start setting up your first M-Base database. For the rest of this article we will assume that you are using M-Base using the sideways RAM option, which is how the program starts up.

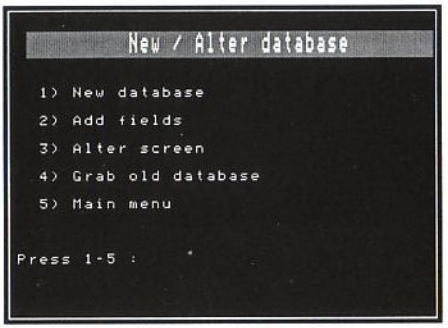
Note that you should not rush ahead and type in your whole book collection at this stage, as there is no save routine supplied in this part of the program. You will have to wait until part 2 to be able to save your databases. Until then you can simply create new databases, add records, view records and extend the database. When you exit from M-Base at this stage you will get an error due to a procedure being missing, yet again this will be supplied next time.

When you run the program you will be presented with a menu of 10 options. To select an option you just need to press the number of that option. To execute an OSCLI command press '*' and then enter the command you wish to execute.

The main menu

SETTING UP A DATABASE

Option 4 of the main menu takes you to a second menu with five options on it. The first of these options is the *New Database* option.



Creating a new database

Once you select this option you will be asked the name of the database. This should be the filename of the database you wish to set up, although it does not need to be.

Next you need to tell the program what fields the database has. You can have a maximum of 24 fields and you need to enter the name of each one and the maximum length that each should require. The first field you enter will be the one which is used to sort the data; enter a blank name and blank length to finish entering fields.

Now you need to tell M-Base where you wish to place each field on the screen when displaying records. Use ZX*? keys to move each field name around the screen and press SPACE to place it. Block characters will show where each placed field's maximum reach is, that is where any data for a given field will reach if it's length is equal to the maximum length set before.

The second option on this menu allows you to extend a database which has already been set up, by allowing you to add fields. First you need to tell the program about the new fields, in a similar way to setting up a new database, and then place all the fields on the screen, again as before. Then the program will make space for all the new fields which may take a few minutes if the database has a lot of records or you are using disc.

The third option allows you to reposition the fields on the screen in new positions allowing you to change the screen display of records. The fourth option on this menu will be described in part two, and the fifth returns you to the main menu.

ENTERING NEW RECORDS INTO THE DATABASE

To enter new records simply select option 2 of the main menu. You will then be shown an empty record with the screen display set up when creating the database. Enter the data to be added to the database and press Return at the end of each field. When you have finished entering one record it will be added to the database and you will be given a new blank entry. To finish adding data press Escape.

This is a good time to describe the powerful text input routine used in this

program. When entering text in most of this program you have full cursor control over the input, rather like a word processor. Pressing left and right cursor keys moves the cursor left and right. Up and down cursor keys move to the start and end of the current line of text. Delete and Copy keys delete behind and at the cursor respectively. Character keys enter that character at the cursor position. Pressing Shift and the Cursor Up key moves back through the fields, and Shift and the Cursor Down key moves down through the fields. Ctrl-U clears the current text line.

VIEWING THE RECORDS

Select option 1 from the main menu. You will then see displayed the first record in the database. That is to say, the first alphabetically to the first field, not the first record entered.

Pressing f0 (or Shift and the Cursor Right key) moves onto the next record, and pressing f1 (or Shift and the Cursor Left key) moves to the previous record. Pressing Escape returns you to the main menu.

While viewing records you can also alter them, simply by typing the alterations, using the method for moving around the record as described in the previous section. The only requirement is that, to make sure that your alterations are stored, you move from that record, e.g. by pressing f0 or f1, rather than pressing Escape.

Whilst viewing records you can use the first of the four searches included in this program. To start the search, press f2 and enter the search string at the prompt. The search routine will then search for the first record which contains that string in its first field. The search is case insensitive; use f3 to continue the search. When no more records are found you will be told so and returned to the last record found.

When each record is displayed the number of that record is displayed at the bottom of the screen. This is the number which should be used when deleting records.

That concludes the first part of the program. In part 2 you will be supplied with features which will allow you to delete records, save and load database files, view information about a database and set up status flags.

```
   10 REM Program M-Base
   20 REM Version B1.60
   30 REM Author  I.J.Palmer
   40 REM BEEBUG  November 1993
   50 REM Program subject to copyright
   60 REM
  100 MODE7
  110 PROCsetup
  120 REPEAT
  130 PROCmenu
  140 PROCoptions
  150 UNTIL O%=0
  160 PROCcleanup
  170 END
  180 :
 1000 DEF PROCsetup
 1010 *SRDATA W
 1020 *SRDATA X
 1030 *SRDATA Y
 1040 *SRDATA Z
 1050 *KEY0 |A
 1060 *KEY1 |B
 1070 *KEY2 |C
 1080 *KEY3 |D
 1090 ON ERROR VDU3:IF ERR=17 OSCLI("FX4
"):GOTO120:ELSE CLS:REPORT:PRINT" at lin
e ";ERL:END
 1100 DIM F$(25),L%(25),T$(25),S$(25),X%
(25),Y%(25),P%(25),C%(50),T%(50),L$(25),
da% &400
 1110 PROCcleardb
 1120 ENDPROC
 1130 :
 1140 DEF PROCcleanup
 1150 PROCclear:IF DC% AND ch%>0 PROCsav
e:ELSE IF NOT(DC%) PROCstinfo
```

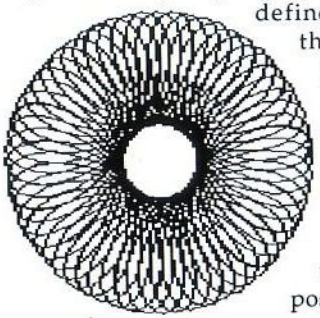
```
 1160 *SRROM W
 1170 *SRROM X
 1180 *SRROM Y
 1190 *SRROM Z
 1200 *BASIC
 1210 ENDPROC
 1220 :
 1230 DEF PROCcleardb
 1240 N$="":N%=0:L%=0:F%=0:E%=0:PR%=0:CS
%=FALSE:FOR A%=1 TO 25:P%(A%)=TRUE:NEXT:
MG%=15:CR%=TRUE:S$="":DC%=FALSE:ch%=0
 1250 ENDPROC
 1260 :
 1270 DEF PROCtitle(P$,Y%)
 1280 P$=CHR$(141)+P$:VDU31,0,Y%,132,157
,135,31,0,Y%+1,132,157,135,31,38,Y%,156,
31,38,Y%+1,156
 1290 PROCcentre(P$,Y%):PROCcentre(P$,Y%
+1)
 1300 ENDPROC
 1310 :
 1320 DEF FNinput(M$,O$,X%,Y%,l%,LA%,HA%
)
 1330 LOCAL OX%,OY%,K%,C%,x%,y%,A%
 1340 OX%=POS:OY%=VPOS:key%=0
 1350 PRINTTAB(X%,Y%);M$;:x%=POS:y%=VPOS
 1360 PRINTO$;TAB(x%,y%);
 1370 M$=O$:A%=4:X%=1:Y%=0:X%=(USR(&FFF4
)AND&FF00)DIV&100
 1380 C%=1:REPEAT:K%=GET
 1390 IF K%=127 AND C%>1 M$=LEFT$(M$,C%-
2)+RIGHT$(M$,LEN(M$)-C%+1):PROCrefresh:C
%=C%-1:VDU8
 1400 IF K%=135 AND C%<=LEN(M$) M$=LEFT$
(M$,C%-1)+RIGHT$(M$,LEN(M$)-C%):PROCrefr
esh
 1410 IF K%=137 AND C%<=LEN(M$) C%=C%+1:
VDU9
 1420 IF K%=21 PRINTTAB(x%,y%);SPC(LEN(M
$));TAB(x%,y%);:M$="":C%=1
 1430 IF K%=136 AND C%>1 C%=C%-1:VDU8
 1440 IF K%=139 C%=1:PRINTTAB(x%,y%);
 1450 IF K%=138 C%=LEN(M$)+1:IF C%>1 PRI
NTTAB(x%,y%);:FOR A%=2 TO C%:VDU9:NEXT
 1460 IF K%>=LA% AND K%<=HA% AND LEN(M$)
<l% M$=LEFT$(M$,C%-1)+CHR$(K%)+RIGHT$(M$
,LEN(M$)-C%+1):C%=C%+1:VDU9:PROCrefresh
 1470 UNTIL K%=13 OR (K%>135 AND K%<140
AND INKEY-1) OR (K%<=11 AND K%>0):OSCLI(
"FX4 "+STR$(X%)):key%=K%:=M$
 1480 :
 1490 DEF PROCrefresh:LOCAL X%,Y%
 1500 X%=POS:Y%=VPOS:PRINTTAB(x%,y%);M$;
"    ";TAB(X%,Y%);:ENDPROC
 1510 :
 1520 DEF PROCcentre(P$,Y%)
 1530 LOCAL X%:X%=19
 1540 X%=X%-LEN(P$)DIV 2:PRINTTAB(X%,Y%)
;P$
 1550 ENDPROC
 1560 :
 1570 DEF PROCclear:LOCAL A%:VDU26,23;11
,0;0;0;0;0;
 1580 PRINTTAB(0,24);SPC(38);
 1590 FOR A%=23 TO 0 STEP-1
 1600 PRINTTAB(0,A%);CHR$(132);CHR$(157)
;SPC(40);:NEXT:PRINTTAB(0,0);"  "
 1610 VDU23;11,255;0;0;0;0;:ENDPROC
 1620 :
 1630 DEF FNlow(I$):LOCAL O$,Q%
 1640 FOR Q%=1 TO LEN(I$):O$=O$+CHR$(ASC
(MID$(I$,Q%,1))OR 32):NEXT:=O$
 1650 :
 1660 REM ***** Read & Write *****
 1670 :
 1680 DEF PROCread(R%):LOCAL A%,P%
 1690 PROCio(TRUE,da%,da%+L%,&400+(R%-1)
*L%):l%=(!da%)AND&FFFF:n%=(!da%)DIV&1000
0
 1700 P%=da%+4:FOR A%=1 TO F%:T$(A%)=$P%
:P%=P%+LEN(T$(A%))+1:NEXT
 1710 ENDPROC
 1720 :
 1730 DEF PROCwrite(R%)
 1740 P%=da%+4:FOR A%=1 TO F%:$P%=T$(A%)
:P%=P%+LEN(T$(A%))+1:NEXT:!da%=(n%*&1000
0)+l%
 1750 PROCio(FALSE,da%,da%+L%,&400+(R%-1
)*L%)
 1760 ENDPROC
 1770 :
 1780 DEF PROCio(r%,s%,f%,p%)
 1790 IF NOT(DC%) AND r% OSCLI("SRREAD "
+STR$~(s%)+" "+STR$~(f%)+" "+STR$~(p%)):
ENDPROC
```

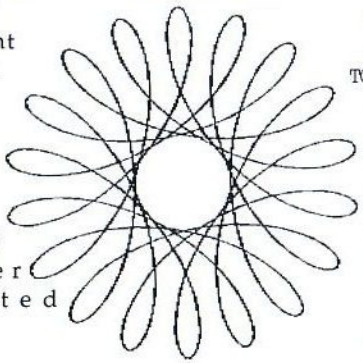
continued on page 48

SpiroPlot

by R.Sterry

This is an attractive little program which generates pictures similar to those produced by a Spirograph. You simply define the size of the 'disc' that you want to use, and the position of the 'pen' within the disc (known as the 'locus' position). Many an hour can be spent playing with this program, trying to produce the magnus opus of computer generated spiroplots.





To get a general idea of what the program can generate, try entering the following values:

Disc Size	Locus
50	1.4
89	1.0
202	0.5
212	1.5
298	1.5

These examples should be enough to trigger your imagination. Happy designing!

Note: This program was originally published in BEEBUG Vol.1 No.9.

```
  10 REM Program SpiroPlot
  20 REM Version B 2.0
  30 REM Author  R. Sterry
  40 REM BEEBUG  Nov 1993
  50 REM Program subject to copyright
  60 :
 100 ON ERROR PROCerror
 110 MODE 7
 120 PRINT TAB(10)"S P I R O P L O T"TA
B(10)STRING$(17,"=")
 130 PRINT TAB(0,12)"Disc Size (0 TO 36
0) "
 140 REPEAT
 150 INPUT TAB(21,12)B
 160 UNTIL B<=360 AND B>=0.01
 170 PRINT TAB(0,14)"Locus Position (0
TO 1.5) "
 180 REPEAT
 190 INPUT TAB(26,14)M
 200 UNTIL M<=1.5 AND M>=0.01
 210 MODE21
 220 VDU 29,639;511;5
 230 VDU 19,1,0,0,0,0,19,0,7,0,0,0
 240 A=-360 : C=A+B : Q=0
 250 REPEAT
 260 X=C*COSQ-M*B*COS(C*Q/B)
 270 Y=C*SINQ-M*B*SIN(C*Q/B)
 280 IF Q=0 MOVE X,Y ELSE DRAW X,Y
 290 Q=Q+B/800
 300 UNTIL FALSE
1000 DEF PROCerror
1010 VDU4
1020 PRINT TAB(0,12)"Try different valu
es (Y/N) "
1030 REPEAT
1040 A$=GET$
1050 UNTIL A$="Y" OR A$="y" OR A$="N" O
R A$="n"
1060 IF A$="N" OR A$="n" THEN CLS:END
1070 RUN
1080 ENDPROC
```

Public Domain

Alan Blundell continues his examination of the latest available PD and shareware for the Beeb.

After last issue, I'm not sure that I can pack in quite so much new software this issue: I've decided to concentrate on two items which I am sure will be of interest.

NECTAR COLLECTOR

Nectar Collector is a Repton Infinity (™) game, written by Alex Card. Alex wrote the game a couple of years ago using the game designing utilities in Repton Infinity, and it was the runner-up in Superior Software's 'Design a Game' competition. Superior had plans at one time to release the game as part of the *Play It Again Sam* series of compilations for the BBC Micro, but as the game needs the full Infinity package to work, the idea was abandoned. Alex later approached Superior for permission to distribute the game as PD, and this permission was given. The game is well worth playing (note, though, that it is only playable if you already have Repton Infinity).

The scenario is not based on the normal 'Repton' character, but on 'Hummy', a hummingbird who has to collect nectar to feed his chicks. In this task, he is harassed by wasps, caterpillars, bats, venus fly traps, spiders and other little nasties. This 'cute' scenario doesn't sound too difficult, perhaps, but Alex says that although the game starts easily, it steadily increases in difficulty. Only one person, he maintains, has so far completed level 8 ... himself!

Alex has moved on to an Archimedes since writing the game, but has included a message with the game that shows a certain fondness for the good old Beeb: "This is for all my Brothers in 6502, hope you enjoy Nectar Collector, please don't send any money but if you can beat all 8 levels you are exceptionally talented. I have moved on from the dear old Beeb but still have fond memories of it and haven't yet found a game to match EXILE on the Archie. Take the Beeb to the year 2000 and beyond. Alex Card."

UTILITIES ROM

Andrew Fiddaman was an early contributor to the availability of PD software for the BBC Micro with his *UtilROM*, an SWR-based collection of utility commands designed specifically for the Master 128 computer. The ROM was not to be a static product, and it has been added to and upgraded over the past couple of years. I have just received the latest upgrade to the ROM, which is available as shareware.

The term shareware means that if you make regular use of the program you should pay a registration fee to Andrew directly. Shareware distribution can be the best of both worlds in some cases, as the user gets to try the program to see if it does suit their needs, and the author gets rewarded for their hard work by the registration fees received (in the case of *UtilROM*, this is £6.50 per user, for as many machines as they use). If there are enough registrations, there is an incentive for the author to produce further programs and to upgrade and enhance their existing work.

UtilROM has evolved into a genuinely useful ROM, with a wide range of commands. The latest release includes new commands together with enhancements to some that were there in the previous release.

Some of the most useful features are:

- Saving & Loading of Basic memory to sideways ROM.

- Ability to set a power-on password as a security aid or anti-theft deterrent

- Sideways ROM saving.

- Disassembler.

- Disc Sector Editor.

In total, the ROM contains 49 star commands, of which some are more useful or uncommon than others. As a flavour, I have picked a few to mention specifically:

*AS - This command is an implementation of the Archimedes SAVE. The program will take the filename from the first line and then save the basic program under this name. The first line must be of the format:

```
REM >filename
```

If it is not, the *UtilRom* will display an error message.

*BVERIFY - This command will check the current Basic program against the file on disc. If the filename is omitted, you will be prompted for it.

*CASE - This command provides an easy method of changing the Caps/Shift lock status from software:

U - Upper Case/Upper Case with Shift
M - Upper Case/Lower Case with Shift
L - Lower Case/Upper Case with Shift

*CMOSDEF - This command returns the CMOS RAM settings to a default state. The optional M parameter causes the configuration to be set to that on p.244 of the Welcome Guide (also *CMOSLOAD & *COMSSAVE).

*COMPACT - This is a replacement for the standard compact command. The difference is that the *UtilRom* first verifies the disc, thus ensuring that no data is moved onto a corrupted sector.

*LPRINT - This command provides an easy way of communicating with your printer. The string (in GSREAD format) is sent directly to the printer. For example, to reset an EPSON compatible printer, the codes required are:

```
<Esc> "@"
```

You could type:

```
VDU 2,1,27,1,ASC"@",3 or:
*LPRINT |[@
```

*PROTECT - This command will prevent the disc in the drive from being catalogued. Although this is by no means a complete protection, it will stop most amateur hackers. Repeating the command will reverse the action (i.e. unprotect the disc).

*TIDYNAMES - This command will rename all files on an ADFS disc so that they begin with an upper case letter with the remainder in lower case.

*WIPE - This adds a *WIPE command to the ADFS.

I have given a fairly long description of this ROM because I think that it contains something that will be useful to everyone, and because some of the facilities are not easily locatable elsewhere.

Andrew has also sent to me several other programs which he has written, including a home accounts package for managing your bank balance, a printer manager including a graphics dump, a RAM filing system which uses SWR banks on the Master, and a utility ROM designed specifically for the assembly language programmer. The last of these makes assembler source code more readable by automatically selecting the right case for different parts of the code.

That's all there is room for this month; next issue, I will describe the latest shareware collection for the Master 512 co-processor and begin another survey of PD libraries and other sources of material for Beeb users as there have been a number of changes since I last did such a survey.

Missile Navigation

This time Cliff Blake isn't shooting at stars.

If you followed the Gravity Orbit programs in BEEBUG Vol.11 No.9 to Vol.12 No.3, you may like to see the incremental technique used in a non-gravity application.

MISSILE COURSES

An unguided missile fired directly towards a moving target, will obviously arrive after the target has moved away. Fitting a homing sensor so that the missile always moves towards the target can result in interception, but the missile may need a greater speed to catch the target. This is called a 'Pure Pursuit Course'. The line between the missile and the target is termed the 'Sight-line', and the axis of a pure-pursuit missile is always aligned with it, so both turn at the same angular rate.

If a homing sensor itself can turn and look at an angle, it is possible for the missile to turn faster and move directly towards the expected interception point. Thus if the sight-line is changing at 0.1 deg/sec, the missile might be adjusting ten times as fast at 1 deg/sec. This is termed 'Proportional Navigation'.

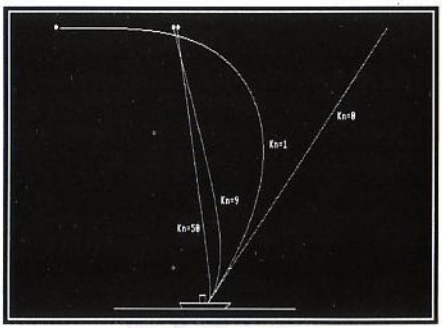
If dAm/dt is the angular rate-of-turn of the missile, and dAs/dt is that of the Sight-line, and Kn is the so-called navigation constant, then the resulting equation is

```
dAm/dt=Kn*dAs/dt.
```

For the unguided case Kn=0, for pure pursuit Kn=1, and for the proportional example above, Kn=10. In most cases a missile would normally be fired at the predicted interception point, and not directly at the target.

THE PROGRAM

This program listed below simulates a ship firing four missiles, with Kn values of 0, 1, 9, and 50, at a hostile aircraft moving from right to left across the top of the screen. This is called a 'Crossing Target' because its direction from the ship is changing. Note that the value of 9 is a big advantage over value 1, but the increase to 50 gains relatively little. On running the program you will get the display shown here. There are no instructions - the simulation runs immediately - but, if you want to experiment with the variables as listed below, you might want to add an input procedure. Press 'Q' to leave the program.



Missile Navigation

Hopefully you will never fire missiles, but the demonstration also suggests that on the football field it is better to close up on an opponent by running for a point ahead of him. However so long as

you have made an approximate allowance, it doesn't have to be too precise.

PROGRAM VARIABLES

The navigation variables adopted are:

Xt,Yt	Target position
Xtv	Target positional steps due to velocity
Xm,Ym	Missile position
Xmv,Ymv	Missile positional steps due to velocity
Rmv	Missile radial step velocity
Xd,Yd	Distance of missile from target
As	Sight-line angle in space
Asp	Previous Sight-line angle
dAs	Sight-line angle increment
Am	Missile angle in space
dAm	Missile angle increment
Kn	Navigation constant

FURTHER EXPERIMENTS

Having carefully saved the original program, alter the Kn value from 0 to 0.5, or some other value less than 1. Change the 50 value to 200, and see the missile tumble due to over-correcting. Try varying the missile velocity Rmv. Although this is a very straight forward simulation it shows how even a small computer can explore important ideas and theories.

```
  10 REM Program MISSILE
  20 REM Version B0.1
  30 REM Author  Cliff Blake
  40 REM BEEBUG  November 1993
  50 REM Program subject to copyright
  60 :
```

```
  100 MODE0:VDU 5
  110 ENVELOPE 1,10,0,0,0,20,20,20,127,0
,0,-20,126,126
  120 PROCship:PROCkn
  130 VDU 4:i%=INKEY(100):VDU 7
  140 REPEAT:g$=GET$:UNTIL g$="Q" OR g$=
"q"
  150 CLS:END
  160 :
 1000 DEF FNangle(x,y)
 1010 LOCAL rd
 1020 IF x>0 AND y=0 =0
 1030 IF x<0 AND y=0 =PI
 1040 IF x=0 AND y>0 =PI/2
 1050 IF x=0 AND y<0 =3*PI/2
 1060 IF ABS(x)/ABS(y)<1E-18 THEN x=SGN(
x)*ABS(y)*1E-18
 1070 rd=ATN(y/x)
 1080 IF x<0 =PI+rd
 1090 IF y<0 =2*PI+rd
 1100 =rd
 1110 :
 1120 DEF PROCcalc
 1130 REPEAT
 1140 Asp=As
 1150 MOVE Xt,Yt
 1160 Xt=Xt+Xtv:Yt=1000
 1170 DRAW Xt,Yt
 1180 MOVE Xm,Ym
 1190 Xm=Xm+Xmv:Ym=Ym+Ymv
 1200 DRAW Xm,Ym
 1210 Yd=Yt-Ym:Xd=Xt-Xm
 1220 As=FNangle(Xd,Yd)
 1230 dAs=As-Asp:dAm=Kn*dAs:Am=Am+dAm
 1240 Xmv=Rmv*COS(Am):Ymv=Rmv*SIN(Am)
 1250 IF ABS(Xd)<10 AND ABS(Yd)<10 THEN
PROChit
 1260 UNTIL hit% OR Ym>1010
 1270 ENDPROC
 1280 :
 1290 DEF PROChit
 1300 hit%=TRUE
 1310 MOVE Xt-12,Yt+12:PRINT"*"
 1320 SOUND 0,1,6,10
```

```
 1330 ENDPROC
 1340 :
 1350 DEF PROCkn
 1360 FOR n%=0 TO 3
 1370 os%=4
 1380 IF n%=0 THEN Kn=0:xo=1105:yo=700:o
s%=0
 1390 IF n%=1 THEN Kn=1:xo=865:yo=600
 1400 IF n%=2 THEN Kn=9:xo=695:yo=400
 1410 IF n%=3 THEN Kn=50:xo=540:yo=300
 1420 PROCinitial:PROCcalc
 1430 GCOL 0,0:MOVE 1279,1000:DRAW Xt+os
%,1000:GCOL 0,1
 1440 MOVE xo,yo:PRINT"Kn=";Kn
 1450 NEXT n%
 1460 ENDPROC
 1470 :
 1480 DEF PROCinitial
 1490 hit%=FALSE
```

```
 1500 Xt=1279:Yt=1000:Xtv=-3
 1510 Xm=650:Ym=20:Rmv=4
 1520 Xd=Xt-Xm:Yd=Yt-Ym
 1530 As=FNangle(Xd,Yd)
 1540 Am=As
 1550 Xmv=Rmv*COS(Am):Ymv=Rmv*SIN(Am)
 1560 ENDPROC
 1570 :
 1580 DEF PROCship
 1590 REPEAT
 1600 READ f$,x,y
 1610 IF f$="M"THEN MOVE x,y
 1620 IF f$="D"THEN DRAW x,y
 1630 UNTIL f$="E"
 1640 ENDPROC
 1650 :
 1660 DATA M,320,0,D,960,0,M,560,0,D,550
,20,D,730,20,D,710,0,M,620,20,D,620,50,D
,640,50,D,640,20,E,0,0
```

Financial Futures

Ian Crawford looks at his retirement prospects with the help of his Beeb.

No matter where you are in your career, retirement and pensions are something you will have been asked to think about at some time. With a vast range of financial packages being offered to us it can be very difficult to cut through the jargon and get some idea of just what our income will look like when the great day comes.

The program *SAVINGS*, listed below, serves as a number of useful 'reminders' to those of us who have not yet done anything about taking out investments. We all know that 'Financial Advisors' (a fancy name for Insurance Salesmen) are constantly pressuring us to save with their company and that we'll all be so much more wealthy if we did, when we retire.

What they quite often fail to mention is that most of the policies promise large sums of money to your surviving family if you die before reaching 60 or 65 years, bit this is not the same sum that you will receive when you survive and retire at 65 and live for another 10-30 years!

Using this program you should be able to ask yourself (and your Financial Advisor) exactly how much cash you will need when you retire, alive, fit and well, at 65. The answer may well surprise you!

One essential detail incorporated in the program is the level of government pension. You can go to *any* DSS office (or write direct to DSS, RPFA Unit, Room 37D, Newcastle Upon Tyne NE98 1YX.) and ask them for a form to enable them to forecast what pension you will receive from the government when you retire at 65 (60 for women). The sum supplied will be as if it was being paid now, even though you may have 10, 20, 30 years to go to retirement.

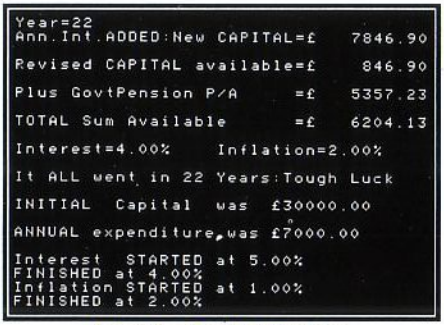
As the government usually increase pensions annually in line with inflation, the figure shown can be used with a certain amount of confidence, particularly if you have only 10-15 years before retirement. However, we're all aware that the government are trying to get younger people to take out private pensions because they can't see themselves being able to fund retirement for those below around 40 years of age in the future.

This is why, when you talk to your Financial Advisor, it is essential to ask about the sum you will receive when you survive and retire, happy in the financial security you always planned for yourself.

When first run, the program has certain defaults set up; you can accept these or enter your own information. The information you can enter is as follows: the total savings available to you, current interest rates, current inflation and annual expenditure (or what you would like to spend!). It assumes that the government pension is at the current level £5471; this is set at line 1140.

When calculating, the program will randomly alter the level of inflation, interest rates and the government pension. All operations are heavily

REMed so you can alter and refine the calculations. These figures are fed back into *PROCcalculate* on a year by year basis until either the money runs out, or you reach the grand old age of 105. The results are then printed out on the screen.



Results of using Savings

I hope the results don't depress you too much. The message from this simple simulation is clear - it's never too early to start planning for your retirement.

```
  10 REM Program SAVINGS
  20 REM Version B1.0
  30 REM Author  Ian Crawford
  40 REM BEEBUG  November 1993
  50 REM Program subject to copyright
  60 :
 100 CLS:MODE7:VDU14
 110 REPEAT
 120 ON ERROR GOTO1000
 130 PROCinit
 140 PROCamount
 150 PROCcalculate
 160 PROCdisplay:wait=GET
 170 PROCcontinue
 180 UNTIL FALSE
 190 END
 200 :
1000 REPORT:PRINT " at line ";ERL:END
1010 :
1020 DEF PROCinit
1030 newcap=0:Year=1
1040 :
1050 int=8:REM Current INTEREST rates
1060 infl=5:REM Current INFLATION level
1070 interest=int:inflation=infl
1080 :
1090 exp=10000:REM ** Assumed ANNUAL su
m covering ALL aspects of EXPENSES when
 Retired **
1100 expenditure=exp
1110 savings=150000:REM Assumed SAVINGS
1120 REM ** To INCLUDE private Pensions
 and Endowment policies and ALL forms of
 Savings you can bring together **
1130 @%=10
1140 govtpen=5471
1150 REM ** Level Of STATE PENSION show
nin `s pa at age 65 (Men)in Authors Case
.Enquire at ANY DSS office for YOUR OWN
 Pension Forecast and entre CORRECT sum
 when you know what it is **
1160 ENDPROC
1170 :
1180 DEF PROCamount
1190 CLS
1200 :
1210 PRINT "Entre TOTAL Savings availab
le to you"
1220 PRINT "OR press <CR> for Default s
um `";savings
1230 PRINT "SAVINGS  must be between"
1240 REPEAT:INPUT "(`20000 Min:`500000
Max)" cap
1250 :
1260 REM * If your Savings are MORE tha
n `500,000.00 then I doubt you've got an
y worries! If it is LESS than `20,000.00
  then I know you have!! **
1270 :
1280 IF (cap>0 AND cap<20000) OR (cap>5
00000) THEN VDU7:PRINTTAB(0,3)SPC(38):VD
U11:REM * Clear incorrect figures and ba
ckspace cursor one line *
1290 UNTIL cap=0 OR (cap>=20000 AND cap
<=500000)
1300 IF cap=0 THEN cap=savings
1310 initcap=cap
1320 CLS
1330 :
1340 PRINT TAB(0,0)"CAPITAL Sum availab
le `";TAB(23,0)cap
```

```
 1350 PRINT "Entre current INTEREST rate
s"
 1360 PRINT "OR press <CR> for Default r
ate of ";int;"%"
 1370 :
 1380 INPUT"Current INTEREST  rate%="int
;:VDU31,24,3:REM ** TAB cursor to X,Y
 1390 IF int=0 THEN int=interest
 1400 PRINT;int
 1410 interest=int
 1420 :
 1430 PRINT''"Entre current INFLATION rat
e"
 1440 PRINT "OR press <CR> for Default r
ate of ";infl;"%"
 1450 INPUT "Current INFLATION rate%="in
fl;:VDU31,24,7:REM ** TAB cursor to X,Y
 1460 IF infl=0 THEN infl=inflation
 1470 PRINT;infl
 1480 inflation=infl
 1490 :
 1500 PRINT''"Now entre what you consider
 to be"''"ALL your  expenses in  retireme
nt"''"OR <CR> for Default sum of `";exp
 1510 INPUT "Annual EXPENDITURE? `"exp;:
VDU31,21,12:REM ** TAB cursor to X,Y
 1520 IF exp=0 THEN exp=expenditure
 1530 PRINT ;exp
 1540 expenditure=exp
 1550 IF exp>cap THEN VDU7:CLS:PRINTTAB(
5,10)"You CAN't afford to retire!":I=INK
EY(300)
 1560 PROCcontinue
 1570 ENDPROC
 1580 :
 1590 DEF PROCcalculate
 1600 CLS
 1610 REPEAT
 1620 PROCadjust
 1630 PROCworkings
 1640 UNTIL (revisedcap+govtpen)<exp OR
Year=40:REM ** If your Retire at 65 and
 live for a further 40 years, Well Done!
 1650 ENDPROC
 1660 :
 1670 DEF PROCadjust
 1680 random=RND(3):REM ** Interest rate
s and Inflation are always changing. Thi
s level (3) is chosen simply to add some
 degree of realism into the calculations
.Experiment with higher figures. It will
 FRIGHTEN you!! **
 1690 random2=RND(4):REM ** This is used
 to Estimate the Govt.State Pension over
 the years. With Govt.trying to REDUCE
 the amount it pays out, the level of an
y increase may NOT equal that of general
 Interest or Inflation rates. **
 1700 IF Year=4 OR Year=6 OR Year=12 OR
Year=17 OR Year=21 THEN int=int-random
 1710 IF Year=5 OR Year=10 OR Year=14 OR
 Year=21 OR Year=27 THEN int=int+random
 1720 IF Year=3 OR Year=7 OR Year=11 OR
Year=19 OR Year=23 THEN infl=infl-random
 1730 IF Year=5 OR Year=9 OR Year=13 OR
Year=18 OR Year=25 THEN infl=infl+random
 1740 IF Year>30 THEN int=infl:REM ** If
 you've lived until 95 yrs then I don't
 reckon you need be too worried about th
edifferences between Interest & Inflatio
nso I've equalised them (at whatever the
y were in year 29) **
 1750 ENDPROC
 1760 :
 1770 DEF PROCworkings
 1780 newcap=((int/100)*cap)+cap:REM ***
*INTEREST Added
 1790 newcap=newcap-((infl/100)*newcap):
REM *** INFLATION Subtracted
 1800 @%=10
 1810 PRINTTAB(0,1)"Year=";Year
 1820 @%=&0002020A
 1830 PRINTTAB(0,2)"Ann.Int.ADDED:New CA
PITAL=`"newcap
 1840 revisedcap=(newcap-exp)
 1850 PRINTTAB(0,4)"Revised CAPITAL avai
lable=`"revisedcap
 1860 newpension=((random2/100)*govtpen)
+govtpen:REM ** RANDOM2 Interest added t
othe Govt.State Pension **
 1870 govtpen=newpension-((random2/100)*
newpension):REM ** RANDOM2 Inflation has
 been taken off the Govt.State Pension *
 1880 PRINTTAB(0,6)"Plus GovtPension P/A
    =`"govtpen
 1890 PRINTTAB(0,8)"TOTAL Sum Available
    =`"revisedcap+govtpen
```

continued on page 20

BEEBUG WORKSHOP

Virtual Arrays

by David Peckett

This month, we return to the theme of the Beeb's shortage of memory. In particular, there is not enough room for very large arrays, such as you might want in a program handling marks or scores, or in a database. However, if you have a disc drive, you can create and use random access files which, in some ways, you can use as slow arrays. These are called "virtual arrays".

Normally, we think of disc files as being for program storage, or as sequential files. Such files contain data written in order by a program, and which another program will read back in exactly the same order. However, a disc system also has the PTR# command, specifying exactly where in a file an item of data is to be written or read. If each item of data is of a known size, then we can go straight to any particular one. For instance, if each data item occupies 20 bytes, then item 100 is 2000 bytes from the start.

VIRTUAL ARRAYS

Suppose that we want a virtual array equivalent to Basic's:

```
DIM array(200,50)
```

The first thing is to find how big a file we need to hold that much data. The array contains 201*51 (numbering starts at zero, remember) floating point (FP) numbers. Since a disc file uses 6 bytes to store an FP number, that means a total of 201*51*6, or 61506 bytes.

To create the file, we need the hex equivalent of that number - use PRINT ~61506 to calculate it. It's &F042, so:

```
*SAVE <filename> 0 +F042
```

will create a file of the correct size. At this stage, the file actually holds the contents of most of the Beeb's memory, but that will be soon overwritten.

Before you can use the virtual array, you must open the file for reading and writing using F%=OPENUP("filename"). Once the file is open, you can write to and read from it with the following routines:

VIRTUAL ARRAY ROUTINES

```
 10000 DEF PROCwritearray(fileno%,x%,y%,
value)
 10010 LOCAL posn%
 10020 posn%=(x%*Ydim%+y%)*6
 10030 PTR#fileno%=posn%
 10040 PRINT#fileno%,value
 10050 ENDPROC

 11000 DEF FNreadarray(fileno%,x%,y%)
 11010 LOCAL posn%,value
 11020 posn%=(x%*Ydim%+y%)*6
 11030 PTR#fileno%=posn%
 11040 INPUT#fileno%,value
 11050 =value
```

Using these routines, the equivalent of array(n1,n2)=value is:

```
PROCwritearray(F%,n1,n2,value)
```

and value=array(n1,n2) is duplicated by:

```
value=FNreadarray(F%,n1,n2)
```

Both cases assume that F% holds the channel number set by OPENUP. If you've used another variable, pass that to the procedure and function.

The keys to the routines are lines 10020 and 11020, which calculate the positions of the appropriate items in the file. The PTR# statement then sets up to read or write that value. The calculations use the global variable *Ydim%*, which holds the number of items along the Y-axis (second dimension) of the virtual array. Set this variable to the correct value - in the example, it is 51 (0-50).

DATABASE FILES

As an alternative, you may want to hold database information. Suppose you run a club and need, say, data about the members' names, addresses, membership numbers and scores in a club competition. It's too much to hold in RAM but you need to get to any item. This is a perfect application for a random-access file.

First, work out how big the file must be. These files only work if every record is the same size - without that, you cannot calculate where any particular record is. Let's allow 20 characters for the name, an address of three 20-character fields and an 8-character postcode, plus an integer membership number and FP score.

The filing system stores strings in their length plus 2 bytes, integers in 5 bytes and FP numbers in 6 (see page 328 of the User Guide). Each record thus has 7 fields and occupies (4*22+10+5+6), i.e. 111 bytes. If we have 200 members, we need a file of 22200 (&56B8) bytes, so create a file for our use with:

```
*SAVE MEMFILE 0 +56B8
```

Once that's done, OPENUP the file and try the database routines. The procedures are similar to the ones for virtual arrays, but transfer data from and to global variables such as *name$* and *membno%*. In practice, of course, you would use your own variables. In use, set *reclen%* to the size of each record (111 in this case). Thus, to get the 32nd record use:

```
PROCreadfile(F%,32,111)
```

Note FNpad and FNdepad. The first forces a string to a fixed size by either truncating it or adding spaces to it, while the second strips trailing spaces from a string. They ensure that all the records are the same size.

DATABASE ROUTINES

```
12000 DEF PROCsavefile
      (fileno%,index%,reclen%)
12010 LOCAL posn%
12020 posn%=index%*reclen%
12030 PTR#fileno%=posn%
12040 PRINT#fileno%,FNpad(name$,20),
      FNpad(add1$,20),FNpad(add2$,20),
      FNpad(add3$,20),FNpad(pcode$,8),
      membno%,score
12050 ENDPROC

13000 DEF PROCreadfile
      (fileno%,index%,reclen%)
13010 LOCAL posn%
13020 posn%=index%*reclen%
13030 PTR#fileno%=posn%
13040 INPUT#fileno%,name$,add1$,
      add2$,add3$,pcode$,
      membno%,score
13050 name$=FNdepad$(name$)
13060 add1$=FNdepad(add1$)
13070 add2$=FNdepad(add2$)
```

```
13080 add3$=FNdepad(add3$)
13090 pcode$=FNdepad(pcode$)
13100 ENDPROC

20000 DEF FNpad(str$,len%)
20010 LOCAL strlen%
20020 strlen%=LEN(str$)
20030 IF strlen%<len% THEN str$=str$
      +STRING$((len%-strlen%)," ")
      ELSE str$=LEFT$(str$,len%)
20040 =str$

21000 DEF FNdepad(str$)
21010 str$=str$+" "
21020 REPEAT
21030 str$=LEFT$(str$,LEN(str$)-1)
21040 UNTIL RIGHT$(str$,1)<>" "
21050 =str$
```

Inevitably, the routines above, and modifications of them, are very much slower than in-memory arrays, hardly surprising when you consider how hard the disc drive is working. For instance, in the first case, random virtual array elements are written at about 1 per sec and read at 2 per sec. Overall throughput, though, is much faster when the records are addressed in order.

Because the filing system uses a part of memory for disc transfers, some virtual array or database accesses may be satisfied directly and speedily from this buffer area and require no physical disc access at that time.

Note, this Workshop was first published in BEEBUG Vol.4 No.6, and has been updated to take account of later developments where BBC micros are concerned. B

Financial Futures (continued from page 17)

```
1900 cap=revisedcap+govtpen
1910 IF int<0 THEN int=int+random:REM *
*As Negative Interest or Inflation are
not likely, this makes sure a positive
rate for both exist **
1920 IF infl<0 THEN infl=infl+random
1930 PRINTTAB(0,10)"Interest=";int;"%"T
AB(18)"Inflation=";infl;"%"
1940 Year=Year+1
1950 REM ** PROCcontinue
1960 REM ** Include PROCcontinue if you
 want to STEP slowly through ALL stages
of the calculations **
1970 ENDPROC
1980 :
1990 DEF PROCdisplay
2000 REM ** VDU2 will Switch ON Printer
2010 REM ** To give you a Hard Copy. Or
you MAY like to entre your own Printer
DUMP routine in here **
2020 @%=10
2030 IF Year<40 THEN PRINT'"It ALL went
in ";Year-1;" Years";":Tough Luck":VDU7
ELSE PRINT'"You outlasted your money by
";Year;" Years!":VDU7
2040 @%=&0002020A
2050 PRINT'"INITIAL Capital was `";i
nitcap
2060 IF Year=40 THEN PRINT'"You've stil
l got `";INT(revisedcap+govtpen);" left!
"
2070 PRINT'"ANNUAL expenditure was `";e
xp
2080 PRINT'"Interest  STARTED at ";inte
rest;"%";'"FINISHED at ";int;"%"
2090 PRINT"Inflation STARTED at ";infla
tion;"%";'"FINISHED at ";infl;"%"
2100 REM ** VDU3 will Switch OFF Printe
r
2110 ENDPROC
2120 :
2130 DEF PROCcontinue
2140 PRINTTAB(5)"Press any key to conti
nue"
2150 REPEAT UNTIL GET
2160 ENDPROC
```
B

Manic Mechanic

by Jonathan Temple

The generator has broken down and the factory is slowly freezing up. Can Manic Mechanic collect all the spanners and reach the generator in time? Find out with this exciting game from Jonathan Temple.

Your objective is to guide the manic mechanic through five different scenarios in order to reach the generator. You'll have to contend with treacherous ice, malfunctioning conveyor belts and some dangerous jumps as you collect the five spanners on each screen. While you're doing this, the time will be continually ticking away - if it reaches zero before you get to the generator, it's bye-bye mechanic.

The moving platforms also deserve a special mention. Instead of carrying you along, like any self-respecting conveyor belt, they'll gladly go off without you, and you will find yourself having to run to stay on them.

The keys to use are the usual 'Z' and 'X' for left and right and Return to jump. In addition, you can pause the game by pressing 'P', and continue with 'C'.

While paused, 'Q' and 'S' will turn the sound on and off respectively; 'F' will finish a game, and 'R' will allow you to re-start a screen - useful if you become trapped, but you do lose a life.

As the program is fairly lengthy, you may need to use a move-down routine, such as the one below, if your computer's PAGE is set higher then &E00.

```
  1 *K.0|F.A%=0TO(TOP-PA.)S.4:A%!&E00=
A%!PA.:N.|MPA.=&E00|MO.|MG.100|M
  2 *FX138,0,128
  3 END
```

Note: This program was originally published in BEEBUG Vol.4 No.5.

```
  10 REM Program Manic Mechanic
  20 REM Version B0.3
  30 REM Author  Jonathan Temple
  40 REM BEEBUG  Nov 1993
  50 REM Program subject to copyright
  60 :
 100 ON ERROR PROCerror
 110 MODE 7
 120 PROCinit:PROCchars
 130 PROCenvs:PROCtitle
 140 REPEAT
 150 MODE 2:PROCsetvars
 160 REPEAT
 170 PROCreset
 180 REPEAT
 190 PROCscreen
 200 REPEAT
 210 PROCman:PROClift
 220 UNTIL E%
 230 IF E%=1 PROCkilled
 240 UNTIL Z%=0 OR E%>1
 250 IF E%=2 PROCnext
 260 UNTIL E%<>2
 270 PROCend
 280 MODE 7
 290 PROChall
 300 UNTIL FALSE
 310 :
1000 DEFPROCman
1010 IFT%>0 T%=T%-1:VDU4:PRINTTAB(6,28)
;T%;"  ":VDU5 ELSE E%=1
1020 A%=X%:B%=Y%:C%=V%:D%=W%
1030 IFINKEY-56 PROCpause
1040 G%=POINT(X%+8,Y%-68):H%=POINT(X%+4
8,Y%-68)
1050 IFINKEY-74 IFJ%+F%=0 IFG%+H%>0 J%=
6:U%=(INKEY-98)-(INKEY-67):SOUND 18,2,10
,5
1060 IFJ% PROCjump:ENDPROC
```

```
 1070 IFG%+H%=0 Y%=Y%-32:W%=W% EOR 1:F%=
F%+1:GOTO1120
 1080 IFF% SOUND 17,1,1,1:SOUND 18,0,0,0
:IFF%>3 E%=1 ELSE IFF% F%=FALSE
 1090 IFG%=8 OR H%=8 IFINKEY-98+INKEY-67
=0 X%=X%+(V%=225)*32-(V%=224)*32:W%=W% E
OR1:GOTO1120
 1100 IFINKEY-98 IFPOINT(X%-8,Y%)<>1 X%=
X%-32:Y%=Y%-(POINT(X%+24,Y%-32)=1)*32:W%
=W% EOR 1:SOUND 18,-10,50,1:IFV%<>225 V%
=225:W%=228
 1110 IFINKEY-67 IFPOINT(X%+72,Y%)<>1 X%
=X%+32:Y%=Y%-(POINT(X%+40,Y%-32)=1)*32:W
%=W% EOR 1:SOUND 18,-10,50,1:IFV%<>224 V
%=224:W%=226
 1120 IFD%<>W% GCOL3,15:MOVE A%,B%:VDUC%
,10,8,D%:MOVE X%,Y%:VDUV%,10,8,W%
 1130 IFPOINT(X%,Y%-20)=3 IFX%MOD64=0 IF
Y%MOD32=28 PROCspanner
 1140 ENDPROC
 1150 :
 1160 DEFPROCjump
 1170 X%=X%+A%(J%)*U%:Y%=Y%+B%(J%)
 1180 J%=J%-1:GCOL 3,15:MOVE A%,B%
 1190 VDUC%,10,8,D%:MOVE X%,Y%
 1200 VDUV%,10,8,W%
 1210 IFY%MOD32=28 IFPOINT(X%+8,Y%-68)+P
OINT(X%+48,Y%-68)>0 J%=0
 1220 IFPOINT(X%,Y%-20)=3 IFX%MOD64=0 IF
Y%MOD32=28 PROCspanner
 1230 IFU%=1 IFPOINT(X%+72,Y%-32)=1 U%=0
 1240 IFU%=-1 IFPOINT(X%-8,Y%-32)=1 U%=0
 1250 IFJ%=0 IFPOINT(X%+8,Y%-68)<1 IFPOI
NT(X%+48,Y%-68)<1 F%=1
 1260 ENDPROC
 1270 :
 1280 DEFPROCspanner
 1290 SOUND 17,4,100,1:K%=K%+1
 1300 IF K%=5 E%=2
 1310 L%=-1:REPEAT L%=L%+1
 1320 UNTIL K%(L%,1)*64=X% AND 1020-K%(L
%,2)*32=Y%
 1330 K%(L%,0)=TRUE:GCOL 3,3
 1340 MOVE X%,Y%:VDU230:PROCscore(100)
 1350 ENDPROC
 1360 :
 1370 DEFPROCpause
 1380 REPEAT N%=GET AND &DF
 1390 IF N%=81 THEN *FX 210,1
 1400 IF N%=83 THEN *FX 210,0
 1410 UNTIL N%=67 OR N%=70 OR N%=82
 1420 IF N%=70 E%=3 ELSE IF N%=82 E%=1
 1430 ENDPROC
 1440 :
 1450 DEFPROClift
 1460 GCOL3,5:MOVE Q%,R%:VDU232,232
 1470 Q%=Q%+IX%:MOVE Q%,R%:VDU232,232
 1480 IFQ%=L1% OR Q%=L2% IX%=-IX%
 1490 ENDPROC
 1500 :
 1510 DEFPROCscore(N%)
 1520 S%=S%+N%:VDU4,17,7,31,6,30
 1530 PRINT LEFT$("00000",5-LEN(STR$(S%)
))+STR$(S%)
 1540 VDU5
 1550 ENDPROC
 1560 :
 1570 DEFPROCkilled
 1580 Z%=Z%-1:SOUND 0,1,100,2
 1590 IF T%=0 VDU19,1,6;0;19,5,6;0;:Z%=0
 1600 TIME=0:REPEAT UNTIL TIME>100
 1610 ENDPROC
 1620 :
 1630 DEFPROCend
 1640 *FX 15,0
 1650 VDU4,28,4,13,15,11,12,26,5
 1660 PROCprint("GAME OVER",352,636,1,3)
 1670 TIME=0:REPEAT UNTIL TIME>200
 1680 ENDPROC
 1690 :
 1700 DEFPROChall
 1710 N%=-1:FOR L%=7 TO 0 STEP -1
 1720 IF S%>S%(L%) N%=L%
 1730 NEXT
 1740 IF N%>-1 PROCcongrats
 1750 PROCdisplay("Hall of Fame")
 1760 PRINTTAB(7,23);CHR$131"Press SPACE
BAR to play";
 1770 REPEAT UNTIL GET=32
 1780 ENDPROC
 1790 :
 1800 DEFPROCnext
 1810 PROCscore(P%*250):P%=P%+1
 1820 IF P%=5 Z%=Z%+1
 1830 IF P%=6 PROCcomplete
 1840 TIME=0:REPEAT UNTIL TIME>200
 1850 ENDPROC
 1860 :
 1870 DEFPROCcomplete
 1880 PROCscore(Z%*500+T%):P%=1
 1890 SB%=SB%-250:T%=SB%:VDU19,8,0;0;
 1900 RESTORE 1960:N%=81:*FX 15,0
 1910 FOR L%=1 TO 10:READ A%,D%:N%=N%+A%
 1920 SOUND 1,-10,N%,D%
```

```
 1930 SOUND 2,-5,N%+48,D%:NEXT
 1940 ENDPROC
 1950 :
 1960 DATA 0,4,8,4,8,4,4,4,8,8,-12,8,4,8
,-12,8,8,8,-16,8
 1970 :
 1980 DEFPROCcongrats
 1990 LOCAL A%,L%,X%,Y%
 2000 FOR L%=6 TO N% STEP -1
 2010 S%(L%+1)=S%(L%):H$(L%+1)=H$(L%)
 2020 NEXT
 2030 S%(N%)=S%:H$(N%)=""
 2040 PROCdisplay("Congratulations!")
 2050 X%=&70:VDU 31,21,6+N%*2
 2060 !&70=&200C0A00:?&74=127:CALL &FFF1
 2070 H$(N%)=LEFT$($&A00,12)
 2080 ENDPROC
 2090 :
 2100 DEFPROCdisplay(M$)
 2110 VDU12,31,4,6,136
 2120 PROClarge(5,0,131,"SCORES")
 2130 FOR L%=3 TO 4
 2140 PRINTTAB((40-LEN(M$))/2-3,L%);CHR$
141;CHR$134;M$
 2150 NEXT
 2160 FOR L%=0 TO 7
 2170 PRINTTAB(5,6+L%*2);L%+1;" ... ";S%
(L%);TAB(17,6+L%*2);"... ";H$(L%)
 2180 NEXT
 2190 ENDPROC
 2200 :
 2210 DEFPROCprint(T$,X,Y,A,B)
 2220 GCOL 0,A:MOVE X,Y:PRINT T$
 2230 GCOL 0,B:MOVE X-8,Y-4:PRINT T$
 2240 ENDPROC
 2250 :
 2260 DEFPROCsetvars
 2270 Z%=3:S%=0:P%=1:SB%=2000:T%=SB%
 2280 FOR L%=9 TO 15:VDU 19,L%,7;0;
 2290 NEXT
 2300 ENDPROC
 2310 :
 2320 DEFPROCreset
 2330 FOR L%=0 TO 4:K%(L%,0)=FALSE
 2340 NEXT:K%=0
 2350 ENDPROC
 2360 :
 2370 DEFPROCinit
 2380 DIM W$(2),A%(6),B%(6),K%(4,2),S%(7
),H$(7)
 2390 C$=CHR$17
 2400 W$(0)=C$+CHR$1+C$+CHR$131+CHR$231
 2410 W$(1)=C$+CHR$6+C$+CHR$132+CHR$232
 2420 W$(2)=C$+CHR$8+C$+CHR$128+CHR$233
 2430 RESTORE2510
 2440 FOR L%=1 TO 6:READ A%(L%),B%(L%)
 2450 NEXT
 2460 FOR L%=0 TO 7
 2470 S%(L%)=4000-L%*500:READ H$(L%)
 2480 NEXT
 2490 ENDPROC
 2500 :
 2510 DATA 0,-32,32,-32,32,0,32,0,32,32,
0,32
 2520 DATA "THE KING","ROCKY IV","JONATH
AN","DANGERMOUSE","COMPO","BEAVER","PENF
OLD","JETSET WALLY"
 2530 :
 2540 DEFPROCscreen
 2550 VDU4,12,19,1,1;0;19,5,5;0;19,8,6;0
;23;10,32;0;0;0;
 2560 PRINTTAB(1,28)"TIME:";T%;TAB(13,28
)"LIVES:";Z%-1'"SCORE:"SPC(7)"SHEET:";P%
;
 2570 PROCscore(0):VDU4,17,3
 2580 RESTORE 2950:IFP%>1 PROCread
 2590 READ N%,K$,L1%,L2%,L3%,W$
 2600 FOR L%=0 TO 4
 2610 K%(L%,1)=ASC(MID$(K$,L%*2+1))-65
 2620 K%(L%,2)=ASC(MID$(K$,L%*2+2))-65
 2630 IFK%(L%,0)=FALSE VDU31,K%(L%,1),K%
(L%,2),230
 2640 NEXT
 2650 FOR L%=0 TO 24 STEP 24
 2660 PRINTTAB(0,L%) STRING$(20,W$(0))
 2670 NEXT
 2680 FOR L%=1 TO 23
 2690 PRINTTAB(0,L%) W$(0);TAB(19,L%) W$
(0)
 2700 NEXT
 2710 FOR L%=1 TO N%*4-3 STEP 4
 2720 A$=MID$(W$,L%,4)
 2730 N1%=EVAL("&"+MID$(A$,3))
 2740 N2%=N1% AND31:N3%=N1% AND32
 2750 N4%=(N1% AND192)DIV64
 2760 D$="":IFN3% D$=CHR$(10)+CHR$(8)
 2770 PRINTTAB(ASC(A$)-65,ASC(MID$(A$,2)
)-65) STRING$(N2%-1,W$(N4%)+D$)+W$(N4%)
 2780 NEXT
 2790 IFP%=5 VDU17,4,17,128,31,6,2,234,2
35,10,8,8,17,136,236,236
 2800 Q%=L1%:R%=L3%:IX%=16
 2810 VDU17,7,17,128,5
 2820 GCOL3,5:MOVE Q%,R%:VDU232,232
```

```
 2830 E%=FALSE:F%=FALSE:J%=0
 2840 X%=128:Y%=956:V%=224:W%=226
 2850 GCOL3,15:MOVE X%,Y%:VDUV%,10,8,W%
 2860 PROCprint("MANIC MECHANIC",192,188
,4,6)
 2870 ENDPROC
 2880 :
 2890 DEFPROCread
 2900 FOR L%=1 TO P%-1
 2910 READ N%,K$,L1%,L2%,L3%,W$
 2920 NEXT
 2930 ENDPROC
 2940 :
 2950 DATA 25,SHCIPLCRSW,320,832,604
 2960 DATA BE03HE03ME44FF41SF01NH23FI02Q
I02GJ02KJ41DK01QJ03BL03BM03RM01BN04PN01S
P01PR42ES41HS25KS83CT81RU02BX10
 2970 DATA 32,QGSKCQCUSV,192,640,636
 2980 DATA CE01FE82JE81ME86SG22RH01CI41F
I82JI81MI08BL01MM41PM81SM01RP42DQ01MQ84B
R01DR02BS45IS01HT03QT82OT01HU04NU42CX01H
V05FW08RW01EX0AQX03
 2990 DATA 25,PCBGBMDTRS,384,896,668
 3000 DATA CE42GE84PE41MF01BI01HI0CBJ02E
K81QL41SN01BO01OO81RO02BP02FP82JP01JQ03O
R42BU24PU83CV23HV82LU82DW22EX21
 3010 DATA 27,FGBILQBTPV,416,800,572
 3020 DATA BE23CE22DE01GE88QE42SH01EI83J
I41MI83RI02QL81BK27CL25DM23EN01QO02OR84L
S01GT81KT03EU81JU04SU01BV02IV05IW05IX0B
 3030 DATA 35,ICRJJUBVRV,256,640,412
 3040 DATA FB2DCE41GE03LE81OE83PG81BH41E
H41JH84GI41DK41IK84OL41BM25GM01RM41GN02K
O23CP22DQ01LQ01MQ82QQ41SS41MT01PT81MU02E
V23HV23MV03DW22IW01LW05CX03IX09
 3050 :
 3060 DEFPROCchars
 3070 VDU23,224,48,56,48,48,32,48,40,40
 3080 VDU23,225,12,28,12,12,4,12,20,20
 3090 VDU23,226,40,56,40,48,32,32,32,48
 3100 VDU23,227,40,56,40,48,40,168,200,1
2
 3110 VDU23,228,20,28,20,12,4,4,4,12
 3120 VDU23,229,20,28,20,12,20,21,19,48
 3130 VDU23,230,0,0,195,126,126,195,0,0
 3140 VDU23,231,223,223,223,0,-5,-5,-5,0
 3150 VDU23,232,0,-1,128,65,34,20,8,-1
 3160 VDU23,233,-1,-1,127,110,98,32,32,0
 3170 VDU23,234,0,0,0,3,63,127,225,237
 3180 VDU23,235,0,0,0,228,242,-3,-4,-2
 3190 VDU23,236,-1,-1,153,-1,153,-1,153,
-1
 3200 ENDPROC
 3210 :
 3220 DEFPROCenvs
 3230 ENVELOPE 1,1,0,0,0,0,0,0,90,-1,-2,
-3,97,97
 3240 ENVELOPE 2,133,8,4,8,3,1,1,126,0,0
,-10,80,0
 3250 ENVELOPE 3,3,-1,-1,-1,70,50,1,100,
-1,-1,-10,101,5
 3260 ENVELOPE 4,133,8,4,8,3,1,1,126,0,0
,-10,126,0
 3270 ENDPROC
 3280 :
 3290 DEFPROCtitle
 3300 VDU 23;10,32;0;0;0;
 3310 FOR L%=1 TO 5
 3320 PROClarge(L%*7-6,1,129,MID$("MANIC
",L%,1))
 3330 NEXT
 3340 PROClarge(1,4,131,"MECHANIC")
 3350 PRINT'TAB(9);CHR$134;"by Jonathan
Temple"
 3360 PRINT'''"ERRORThe keys to use are
as follows:"''"LINE'Z' and 'X' to go lef
t and right,"''"LINEand the Return key to
 jump."''
 3370 PRINT'"MODPress any key to start."
:IFGET
 3380 ENDPROC
 3390 :
 3400 DEFPROClarge(V%,W%,C%,M$)
 3410 LOCAL A%,B%,L%,M%,S%,X%,Y%
 3420 FOR L%=1 TO LEN(M$)
 3430 ?&70=ASC(MID$(M$,L%)):?&79=0
 3440 X%=&70:Y%=0:A%=10:CALL &FFF1
 3450 FOR Y%=0 TO 6 STEP 3
 3460 VDU 31,V%,W%+Y%/3
 3470 IF L%=1 VDU C%+16,154
 3480 FOR X%=0 TO 6 STEP 2:S%=160
 3490 FOR Z%=0 TO 2:M%=?(&71+Y%+Z%)
 3500 A%=4^Z%:B%=2^(Z%*3)-(Z%=0)
 3510 IF (M% AND 2^(7-X%)) S%=S%+A%
 3520 IF (M% AND 2^(6-X%)) S%=S%+B%
 3530 NEXT:VDU S%:NEXT,
 3540 V%=V%+4-2*(L%=1):NEXT
 3550 ENDPROC
 3560 :
 3570 DEF PROCerror
 3580 MODE7:REPORT:PRINT" at line ";ERL
 3590 *FX15,1
 3600 END
 3610 ENDPROC
```

Census (Part 4)

Paul Goldsmith rounds off his suite of census programs with a cross tabulation facility.

Having collected your data and looked at the frequency distributions of the answers, you can get even more out of your survey by cross tabulating the responses to one set of questions against another. You might look at age against the sort of goods bought, or origin against the amount spent. With, say, 10 questions on the form it would be possible to make 45 cross tabulations, although not all of them will be meaningful.

The program CROSSDU is listed below and does the necessary maths, enter it, add the procedures window1, 2 and 3, t and blue from the first article (BEEBUG July) then save it under the same name. It can now be selected from the Analysis Menu option 3 and will create up to 5 cross tabulations at a time.

The program first asks for a character to distinguish a new set. This character must be one which can be included in a file name for the disc filing system such as A, B, C, 1, 2, 3, etc. Upper and lower case letters are treated the same as is usual. The results are saved as <Code>DA + the distinguishing character. In the example we have been working on this will be SHOPDAA if the distinguishing character is 'A'. The program then asks for the number of cross tabulations you want to include in the set; you can have up to 5. The next screen asks for the Independent (across the page) question number for the first cross tabulation, and then when this has been entered the Dependent (down the page) question number. You will then be asked for the next pair and so on until the entries are complete.

Next, the program reads the records and calculates the results for the selection, keeping you informed by displaying the record numbers as it does so; missing records are skipped. When the calculations are complete and saved a beep indicates this and a message confirms completion.

You will be returned to the Analyse Menu where you may make another selection with a different distinguishing character, again using option 3. However, it is more likely that you will want to see the results of the first one by selecting option 4 from Analyse to display the results. To use this you need to type in the second program, CROSSDI, as listed below. Again you need to add the procedures as above and, again, it must be saved under the name CROSSDI.

The Cross Tabulation Display Menu first asks for the character which distinguishes the set of cross tabulations. When loaded the menu displays the question numbers and the keys 'A' to 'E'. First select the one required by pressing one of these then 'F' or 'G' to view on the screen or for a print out. The screen display as included in the program only gives an idea of the results. If you are lucky enough to have a B+ or a Master some minor alterations to the program to alter the modes can enable you to use MODE 0 or 128. If you have OVERVIEW, inserting a line with *WIDE before the mode change will enable even more to be included on the screen. The program adjusts the printed output on an Epson

compatible printer to fit an A4 page. The results include the coincident totals, the totals across, down, the grand total and the percentages of the totals across, down and of the grand total.

I have written the program so that the percentage of each result is calculated in relation to the total of responses across, down and of the grand total. It must be understood that these do not add up to 100% in cases where respondents have given more than one reply to a question or have not replied to a question.

The Census Suite of Programs has been used on a number of occasions over several years for practical research, particularly in the field of ascertaining the needs of the elderly in the community. It is flexible enough to be adapted for virtually any questionnaire and, since the data is in a simple and uncomplicated form its manipulation in standard spreadsheet packages is very easy. I hope you find many good uses for it.

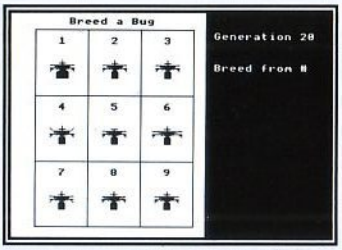
```
  10 REM Program Crossdi
  20 REM Version B 1.0
  30 REM Author  Paul Goldsmith
  40 REM BEEBUG  November 1993
  50 REM Program Subject to Copyright
  60 :
 100 ONERROR VDU7:REPORT:PRINT" at line
 "ERL:END
 110 VDU26:PROCcode:DIMZ%(5,H%,H%),T%(J
%),A%(J%),QI%(5),QD%(5),A$(J%,H%),SI%(H%
),SD%(H%)
 120 VDU15,26:MODE 135:PROCt(3,"Crossta
bulation Display"):REM MODE 7 for BBC B
 130 PRINTTAB(3,20)"AND PRESS RETURN":P
RINTTAB(10,6)"SELECTION OF SURVEY"TAB(13
,7);"AND BASIC DATA"
 140 PROCgetparam:*FX202,32
 150 INPUTTAB(3,14)"Type the character
which"''TAB(3,15)"distinguishes the cross
tab "S$:IF S$="" GOTO 150
 160 PROChead
 170 PROCget
 180 :
 190 REM MENU*********
 200 PROCwindow1:CLS:PRINTTAB(5,0);"CRO
SSTABULATION SELECTION"
 210 FORQ%=1TO Numb%:PRINTTAB(3,Q%+1);Q
I%(Q%)TAB(10,Q%+1);"by"TAB(15,Q%+1);QD%(
Q%)TAB(30,Q%+1);CHR$(Q%+64):NEXT:Q%=0
 220 PRINTTAB(3,8);"Single crosstab to
screen";TAB(30,8);"F"
 230 PRINTTAB(3,9);"Single crosstab to
printer"TAB(30,9);"G"
 240 PRINTTAB(3,10);"All crosstabs to p
rinter"TAB(30,10);"H"
 250 PRINTTAB(3,11);"Quit"TAB(30,11);"I
"
 260 PRINTTAB(3,12);"Another selection
?"TAB(30,12);"J"
 270 PRINTTAB(3,14)"Select Crosstab the
n option"
 280 *FX15,1
 290 A=GET
 300 IF A>74 OR A<65 VDU7:GOTO 280
 310 IF A<70 K%=A-64:QI%=QI%(K%):QD%=QD
%(K%)
 320 IF A=70 AND K%>0 N%=K%:GOTO 390
 330 IF A=71 AND K%>0 N%=K%:PROCprint
 340 IF A=72 PROCprint
 350 IF A=73 THEN CHAIN"ANALYSE"
 360 IF A=74  RUN
 370 IF K%=0 VDU7:PRINTTAB(2,20)"Select
 crosstab first ":TIME=0:REPEATUNTILTIME
=100
 380 GOTO 190
 390 MODE4:PROCdisplay:MODE 7:PROCt(3,"
Crosstabulation Display"):GOTO 190:REM m
odes 128 and 135 for Master
 400 END
 410 :
1000 DEF PROCquest
1010 ONERROROFF:B=OPENIN G$:PROCefile(B
,G$)
1020 INPUT#B,Head$,Q%:FOR X%=1 TO Q%:IN
```

```
PUT#B,T%:A%(X%)=T%:FORY%=0 TO A%(X%):INP
UT#B,A$(X%,Y%):NEXT:NEXT
 1030 CLOSE#B:ENDPROC
 1040 :
 1050 DEF PROCcode
 1060 ONERROROFF:C%=OPENIN"CODE":PROCefi
le(C%,"CODE")
 1070 INPUT#C%,F$,J%,H%,Rmax%
 1080 D$=F$+"DA":H$=F$+"HED":G$=F$+"QUE"
:Code$=F$+"*****":P$=F$+"PAR"
 1090 CLOSE#C%:ENDPROC
 1100 :
 1110 DEF PROChead
 1120 ONERROROFF:C%=OPENINH$:PROCefile(C
%,H$)
 1130 INPUT#C%,Head$,J%
 1140 FORT%=1TOJ%
 1150 INPUT#C%,A%(T%)
 1160 FORY%=0TOA%(T%):INPUT#C%,A$(T%,Y%)
 1170 NEXT:NEXT
 1180 FORT%=1TOJ%:A%(T%)=A%(T%)-D%:NEXT
 1190 CLOSE#C%
 1200 ENDPROC
 1210 :
 1220 DEF PROCget
 1230 E$=D$+S$
 1240 ONERROROFF:C%=OPENINE$:PROCefile(C
%,E$)
 1250 INPUT#C%,Numb%:FORQ%=1TONumb%:INPU
T#C%,QI%(Q%),QD%(Q%)
 1260 FORY%=1TOH%
 1270 FORX%=1TOH%
 1280 INPUT#C%,Z%(Q%,X%,Y%)
 1290 NEXT:NEXT:NEXT
 1300 CLOSE#C%
 1310 ENDPROC
 1320 :
 1330 DEF PROCcalc
 1340 FORY%=1TOA%(QD%):FORX%=1TOA%(QI%)
 1350 SI%(X%)=SI%(X%)+Z%(N%,X%,Y%):SD%(Y
%)=SD%(Y%)+Z%(N%,X%,Y%)
 1360 NEXT:NEXT
 1370 Tot%=0
 1380 FORX%=1 TO A%(QI%):Tot%=Tot%+SI%(X
%):NEXT
 1390 VDU7
 1400 ENDPROC
 1410 :
 1420 DEF PROCdisplay
 1430 IF A%(QI%)>14 BIG=TRUE ELSE BIG=FA
LSE
 1440 :
 1450 PROCcalc:@%=&00008:IF BIG=TRUE @%=
&00006
 1460 VDU14:PRINT'TAB(2);Head$'TAB(5);F$
;" CENSUS"'TAB(3);"CROSS TAB  Q ";QI%;"
BY Q ";QD%;TAB(3);A$(QI%,0);" BY ";A$(QD
%,0):PRINTTAB(3);"Number of forms return
ed ";Forms%
 1470 PRINTTAB(3)"Lowest  Form number ";
Start%:PRINTTAB(3)"Highest Form number "
;End%
 1480 PRINTTAB(2);:PROCdate
 1490 IF BIG PRINT':PRINTTAB(13);:FORX%=
1TOA%(QI%):PRINT X%;:NEXT:PRINT" "
 1500 IF NOT BIG PRINT':FORX%=1TOA%(QI%)
:PRINTTAB(7+8*X%);A$((QI%),X%);:NEXT:PRI
NT" "
 1510 FORY%=1TOA%(QD%)
 1520 PRINT'TAB(2);A$(QD%,Y%);TAB(12);":
";:FORW%=1TO A%(QI%):PRINTZ%(N%,W%,Y%);:
NEXT:PRINTSD%(Y%):PRINT" "
 1530 IF BIG @%=&20105 ELSE @%=&20107
 1540 PRINTTAB(11);"a:";:FORW%=1TO A%(QI
%):IF SD%(Y%)  PRINT100*Z%(N%,W%,Y%)/SD%
(Y%);"%"; ELSE PRINT 0;"%";
 1550 NEXT
 1560 IF SD%(Y%) PRINT 100*SD%(Y%)/Tot%;
"%"; ELSE PRINT 0;"%";:PRINT" "
 1570 PRINTTAB(11);"d:";:FORW%=1TO A%(QI
%):IF SI%(W%) PRINT100*Z%(N%,W%,Y%)/SI%(
W%);"%"; ELSE PRINT 0;"%";
 1580 NEXT:PRINT" "
 1590 PRINTTAB(11);"t:";:FORW%=1TO A%(QI
%):IF Tot% PRINT 100*Z%(N%,W%,Y%)/Tot%;"
%"; ELSE PRINT 0;"%";
 1600 NEXT:PRINT" "
 1610 IF BIG @%=&00006 ELSE @%=&00008
 1620 NEXT
 1630 PRINT'TAB(12);:FOR X%=1TOA%(QI%):P
```

continued on page 46

Best of BEEBUG

Applications I Disc



Business Graphics - for producing graphs, charts and diagrams
Video Catalogue - 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.

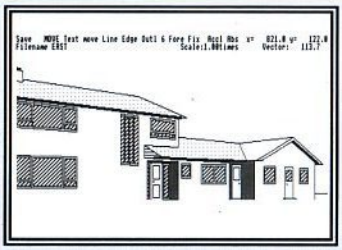
The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical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

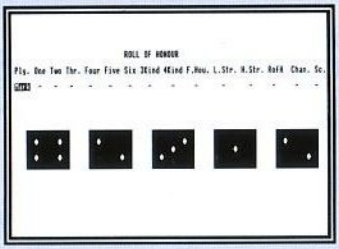
	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£5.95	**ASTAAD** (3.5" ADFS)	1408a	£5.95
Applications II (80 track DFS)	1411a	£4.00	**Applications II** (3.5" ADFS)	1412a	£4.00
Applications I Disc (40/80T DFS)	1404a	£4.00	**Applications I Disc** (3.5" ADFS)	1409a	£4.00
General Utilities 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



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

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

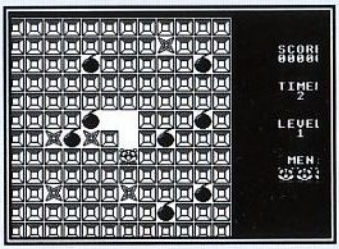
Julia Sets - fascinating extensions of the Mandelbrot set

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		Stock Code	Price
File Handling for All Book	BK02b	£9.95			
File Handling for All Disc (40/80T DFS)	BK05a	£4.75	**File Handling for All** Disc (3.5" ADFS)	BK07a	£4.75
Joint Offer book and disc (40/80T DFS)	BK04b	**£11.95**	**Joint Offer book and disc** (3.5" ADFS)	BK06b	**£11.95**
Magscan (40 DFS)	0005a	£9.95	**Magscan Upgrade** (40 DFS)	0011a	£4.75
Magscan (80T DFS)	0006a	£9.95	**Magscan Upgrade** (80T DFS)	0010a	£4.75
Magscan (3.5" ADFS)	1457a	£9.95	**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

Wordwise User's Notebook: On the Transfer List

David Polak shows how to create strings in Basic for use in Wordwise Plus.

Basic programs often produce some output which can be used in other environments such as Wordwise Plus. The *Spool facility can be helpful in solving the problem of how to transfer the material. One alternative method was discussed in Wordwise User's Notebook in BEEBUG Vol.11 No.6. Another possibility is to include the material to be transferred in a form which can be loaded into a Wordwise Plus Segment, and to read it selectively into the Wordwise Plus text.

XBasWW+

The XBasWW+ program listed here enables up to ten quite long strings, i.e. phrases, numbers or a mixture, prepared in Basic, to be sorted if required, and automatically transferred to a Segment in Wordwise Plus from which they can be called as needed.

OPERATION

Type in the Basic program XBasWW+ and save it. Prepare the !BOOT and the !B sequences as shown; these can be typed in Wordwise Plus and saved as !BOOT and !B respectively, or by using the *BUILD method. Don't forget *OPT 4 3 to enable the !BOOT routine. Go to Wordwise Plus, type and save the SEG2 program.

Now from the Wordwise Plus menu, or from Basic, invoke XBasWW+ by *EXEC !B. Choose the number of phrases in response to the request. Enter each phrase and press Return. When all have been entered the program automatically takes you into Wordwise Plus and loads Segments 0 and 2. Type a few sentences and save them from the Wordwise Plus menu as "text".

Segment 0 now holds the strings (each with an identification letter), while Segment 2 holds the Wordwise Plus program SEG2 which operates when called. Do this by keying Shift-f2.

On request, identify a string by letter, which when keyed brings you back to TEXT with the string inserted at the cursor and followed by a space. Should your identifier be outside the range, the "sorry" section of SEG2 brings you back to TEXT with the message "Press <DELETE>". Yes, you can call the identifier by upper or lower case letter, and thank you to Mr. Toad for that!

RANGE

As presented, 10 strings can be displayed. Should you wish to have more than 26 strings, they will be distinguished by both upper and lower case and other, strange, identifiers. Also you will need to modify the program to display more than ten strings.

QUOTATION MARKS

These can be omitted at the start and end of the string unless you require the string to begin with one or more spaces or to end with more than one.

SORTING

A choice is given between alphabetical and numerical sorting. A screen warning is given to avoid using numerical sorting which rubs out alphabetical strings.

BOOT TO RE-USE STORED STRINGS

The strings are held in file "Z" and will remain on disc to be called up using Shift-Break. If you wish to make another set of strings, then BEFORE doing so, rename the file for example:

```
RENAME Z STRING1
```

and recall when required with:

```
RENAME STRING1 Z
```

otherwise Z will be overwritten and lost when you use !B. Use *EXEC !B to prepare a new set of strings.

```
!BOOT
REM.....!BOOT
*WORD. :SEL.SEG.0
:LOAD TEXT "Z"
:SEL.SEG.2
:LOAD TEXT "SEG2"
:SEL.TEXT
:LOAD TEXT "test"
:DIS.

!B
REM.....!B
*B.
CH."XBasWW+"

SEG2
REM.....SEG2
SEL.SEG.0
C.T.
DIS.
P." Select the Key-letter"
REM alternatively Q$=GCK$+") "
Q$=CHR$(ASC GCK$ AND &DF)+") "
C.T.
FIND Q$
IF EOT THEN GOTO sorry
C.RIGHT 3
Q$=GLT$
SEL.TEXT
TYPE Q$
DISP.
END

.sorry
SEL.TEXT
DIS.
TYPE"#"
VDU 7,31,1,0
P." Character "+Q$+"is NOT RECOGNISED ..."
VDU 132,157,134
P."..... Press <DELETE> "
```

```
  10 REM Program XBasWW+
  20 REM Version B.1
  30 REM Author David Polak
  40 REM For ADFS only
  50 REM BEEBUG November 1993
  60 REM Program subject to Copyright
  70 :
 100 MODE7
 110 PROCnumber
 120 PROCcontents
 130 PROCsort
 140 PROCspool
 150 OSCLI "*EX.!BOOT"
 160 END
 170 :
```

```
 1000 DEF PROCnumber
 1010 PRINTCHR$(131);:INPUT" How many st
rings? "X%
 1020 DIMA$(X%),A(X%)
 1030 ENDPROC
 1040 :
 1050 DEF PROCcontents
 1060 FOR X=1 TO X%
 1070 PRINT"Number ";X;" please:- "
 1080 INPUTA$(X)
 1090 NEXT
 1100 FOR X=1TO X%:PRINTA$(X):NEXT
 1110 ENDPROC
 1120 :
 1130 DEF PROCsort
 1140 REPEAT
 1150 PRINT'CHR$(131);"SORT THESE?";
 1160 IF FNagree=FALSE ENDPROC
 1170 PRINTCHR$(131);"SORT ALPHABETICALL
Y?";
 1180 IF FNagree=TRUE PROCstringSort ELS
E PROCnumSort
 1190 FOR X=1TO X%:PRINTA$(X):NEXT
 1200 PRINT">>> OK?";: UNTIL FNagree=TRU
E
 1210 ENDPROC
 1220 :
 1230 DEF PROCstringSort
 1240 REPEAT:F=0
 1250 FOR X=1TOX%-1
 1260 IFA$(X)>A$(X+1):T$=A$(X):A$(X)=A$(
X+1):A$(X+1)=T$:F=1
 1270 NEXT
 1280 UNTIL F=0
 1290 ENDPROC
 1300 :
 1310 DEF PROCnumSort
 1320 PRINT'CHR$(129);"NB 'NUMERICAL SOR
T' DELETES WORDS"
 1330 PRINT'CHR$(131);"Answer 'Y' for Nu
merical Sort";
 1340 IF FNagree=FALSE ENDPROC
 1350 FORX=1TOX%:A(X)=VAL(A$(X)):NEXT
 1360 REPEAT:F=0
 1370 FOR X=1TOX%-1
 1380 IFA(X)>A(X+1):T=A(X):A(X)=A(X+1):A
(X+1)=T:F=1
 1390 NEXT
 1400 UNTIL F=0
 1410 FORX=1TOX%:A$(X)=STR$(A(X)):NEXT
 1420 ENDPROC
 1430 :
 1440 DEF PROCspool
 1450 *SP.Z
 1460 PRINT"":PRINT""
 1470 FOR X=1TO X%
 1480 PRINT CHR$(X+64);")";A$(X):NEXT
 1490 *SP.
 1500 ENDPROC
 1510 :
 1520 DEF FNagree
 1530 PRINT CHR$(134);" Y/N ":IF CHR$(AS
C GET$ AND &DF)="Y"=TRUE ELSE=FALSE
```

Procedures and Functions

Marshal Anderson serves up a dish of these delights.

Basic is chock full of very wonderful commands, and BBC Basic is particularly blessed, but it can't provide everything you need in the form of a simple command word. Take INPUT for example; this is quite a sophisticated command, you can add a text message to it, it gives the user access to the screen editor, and it returns a string or number that you can use in the program. Unfortunately, you usually want more than that, you may want to accept only numbers, strings of a certain length, convert the input to capitals and so on.

One of the purposes of functions and procedures is to let you write your own commands that do exactly what you want and can be used at any time. Let's start by looking at the technicalities. A procedure is a section of code that is given a label or name that looks like this:

```
100 DEF PROChello
```

You can read this as 'define a procedure called *hello*'. Having given it a label you can now put in your code; just to get going add:

```
110 PRINT "Hello"
```

Having added the code we need to finish off with:

```
120 ENDPROC
```

You can read this as 'end procedure'. Now add the following code (this would be part of your 'main' program):

```
10 PROChello
20 END
```

When you run the program you will get 'Hello' on the screen. What's happening is that *PROChello* tells Basic to branch to that procedure from line 10, execute the code within the procedure, and then return to the point from which it branched. Okay, not too spectacular, but the point is that you have given Basic a new command.

To digress for a moment and talk about style, you'll note that I've used lower case for the procedure name (DEF PROC must be in capitals and there mustn't be a space after PROC). This is for readability, helping the name of the procedure stand out and is good practice (and you know it). Also, you cannot have spaces in a procedure name.

Your new command, however, can be a lot more complex than this example; the code can be as long as you like, and there are no restrictions as to the contents of a procedure with one single exception: for a reason no one has ever satisfactorily explained, you cannot use the MODE command within a procedure.

GETTING CLEVER

There are a couple of extra goodies that can help make your procedures even more powerful. The first is passing *parameters*. You can follow your initial labelling of a procedure with a list of parameters that will have values passed to them; try this:

```
10 PROCmulti("+",5)
20 STOP
100 DEF PROCmulti(A$,N%)
110 FOR X%=1 TO N%
120 PRINT A$;" ";
130 NEXT
140 PRINT
150 ENDPROC
```

Here the definition of the procedure *PROCmulti* has two parameters attached to it; *A$* and *N%*, separated by a comma. When the procedure is called (line 10) the parameters are replaced by values (called arguments) in the brackets following the call. These arguments must match the types the procedure expects from its definition or you will get the error 'Arguments at line...' *PROCmulti* takes a string and prints it out a number of times. The string and the number of times it is printed will depend on the arguments you provide when you call the procedure; try changing the arguments at line 10 to see the effect. These arguments can themselves be variables, note that the variables referred to in the actual procedure code are named from the DEF PROC line, their values are provided by the procedure call.

The second sophistication is the use of *local* variables. Actually, you have used them already - *A$* and *N%* are automatically made local variables as they are the formal parameters of the procedure; that is they only exist within the procedure. Although this initially seems rather perverse it is a jolly good idea. It allows you to write procedures containing variables that will not affect, or be affected by, anything you write in the main program or in other procedures. It does not, however, apply to the variable X% in line 110, this is a *global variable* and could cause havoc if used elsewhere in your program. To prevent this we can use the command LOCAL; adding LOCAL X% at line 105 would isolate the X% in your procedure from any other X% in the rest of the program.

A PRIVATE FUNCTION

Functions are very similar to procedures; the difference is that they *return a result*. Obviously it is possible to *get* a result from a procedure by assigning the result you want to a global variable within the procedure, but a function does it far more neatly and avoids the need for global variables for this purpose. Here's a short example which finds the average of three numbers. First we name our function:

```
100 DEF FNaverage(A,B,C)
```

The same rules apply to the function name as to procedures. Now we put in the code:

```
110 LOCAL result
120 result=(A+B+C)/3
```

And then we finish off with a rather strange dangling '=' sign.

```
130 =result
```

This is how functions must finish, though it looks very odd and might take some getting used to. Now we can write some code to make use of the function. This part of it works in exactly the same way as things like SQR(); you deal with it as a value, you don't have to explicitly tell Basic to do the function.

```
10 INPUT X,Y,Z
20 PRINT FNaverage(X,Y,Z)
30 END
```

Functions don't have to have attributes; they can do some very simple but useful jobs without them. For example, here is a really useful function that finds its way into all my programs.

```
100 DEF FNyn
110 LOCAL A$
120 PRINT "Press Y or N"
130 REPEAT
140 A$=GET$
150 IF A$="n" THEN A$="N"
160 IF A$="y" THEN A$="Y"
170 UNTIL A$="N" OR A$="Y"
180 =A$
```

The advantage of this is that it produces only the results I want to deal with and can be used in code like:

```
200 PRINT "Continue?":IF FNyn="N" THEN
END
```

The advantage of this kind of procedure is that, once it's written and fully tested, you can use it in all your programs; many programmers keep a file of procedures which they just spool in when they need them.

ERRORS

Apart from the attributes error mentioned earlier, which produces a helpful error message, the most common error with functions and procedures is unexpectedly dropping in and out of them, which doesn't. You can find yourself in a real mess with this, take out the ENDs in the above examples to see what I mean. The problem is that, if Basic encounters a DEF statement in the normal stream of the program, it just ignores it, so you can find a procedure being executed when you didn't expect it. Further, Basic doesn't know how long a procedure should be so, if there is no ENDPROC or dangling '=' or it is bypassed for some reason, Basic will just execute the next bit of code after the procedure. The best way of avoiding this is to put all your functions and procedures at the end of the main program with an END command before them. Most of the listings in BEEBUG work exactly like this.

BUT THERE'S MORE

Functions and procedures don't just affect programming in the technical way described above, they can also affect the *way* we program. Before the introduction of these parts of the language, the best Basic had to offer in this line was the subroutine (GOSUB); local variables, use of names, the passing of information were not available. Programs tended to use vast numbers of GOTOs which made the development and de-bugging of programs very difficult because it was hard to trace exactly where in the program a problem was.

Functions and procedures encourage a quite different approach. Instead of writing a program from line 10 onwards it allows you to develop the program as small modules (the smaller the better) each of which is tested as it is developed. The use of local variables cuts down the problems you might have keeping track of global variables and the way they might affect each other. Further, when something goes wrong it's usually much easier to hunt it down and fix a particular procedure because procedures are so easy to identify.

When you start developing a program you can use your listing as a planner for what you need to do. Below is an example from the July issue which shows how a whole game is summed up in a few lines.

```
      100 MODE 1
      110 PROChello
      120 PROCinitialise
      130 PROCstartnewgame:CLS
      140 PROCwhereamI
      150 PROCanynews
      160 IF FNwhatnow="M" PROCmove ELSE
PROCshoot
      170 PROChowamI
      180 IF PLAYING% GOTO 140
      190 ON FNished GOTO 130,200,210
      200 PLAYING%=TRUE: YOUAREHERE% =
HAZARD%(6): CLS:GOTO 140
      210 PRINT'''"Byeeeeeeeee"
      220 END
```

You can start your program like this (though you might try to use less GOTOs) and then develop the procedures one at a time, top-down programming, or start by writing the procedures and then assemble them into the complete program, bottom-up. You will probably end up doing a combination of both but you will certainly end up in more control of your code.

BEEBUG Education

by Mark Sealey

Last July, BEEBUG Education looked at the first of two reports published recently on the rôle and impact of computers in education. A somewhat depressing picture was painted in the findings of a Department for Education (DfE) publication, the Statistical Bulletin 6/93, 'Survey of Information Technology in Schools'. This publication limited itself to broadly factual (numerical) information on the availability and use of IT - much of which is still BBC 8-bit hardware (see BEEBUG Vol.12 Issue 3).

Perhaps even more interesting than this is material to appear from the academic world, the Centre for Education Studies at Kings' College London. It evaluates the impact of Information Technology on both primary and secondary pupils' achievements in the curriculum. The report of the Kings' team is called, in fact, 'The ImpacT Report' and has been compiled by three of the most nationally respected academics in the field. Does it make happier reading for those of us who have held to our beliefs in the validity of IT in the classroom and made substantial use of BBC machines and their software?

As in July's discussion of the DfE publication, not all the data refers explicitly to the Acorn 8-bit hardware. Since, however, the BBC micro has been the platform on which much of the pupils' work analysed in the course of these investigations has been done, it is relevant material for those using it.

For the record - and as one would expect from a body like Kings' - the research methodology is explained in the early chapters of the ImpacT report. It began in January 1989, concentrated on individual classes, rather than whole schools, and was conducted with reference to four curriculum areas: English, geography, mathematics and science. Significantly, the study was 'longitudinal'; that is, the same pupils' progress was measured over time, rather than as a snapshot in one month, term or year.

Perhaps more significant still, is that progress was measured against specific learning skills, including, particularly, the effect of IT on what educationalists call "higher level thinking". Those are skills requiring abstraction, use of inference and deduction, creativity and decision-making rather than mere use of memory and 'knowledge'. Many of the pioneer advocates for IT in education have long maintained that this is a major, but overlooked, area in which IT has a contribution. Such an emphasis by the Kings' team is hence particularly exciting.

Over 2,000 pupils were involved from 19 Local Education Authorities (LEAs); these were mainly in the south of England and arranged in "matched pairs"! One of each pair was in a class that made "high" use of IT, the other from a class making less use of IT.

The pupils studied were aged 8-10, 12-14 and 14-16 years. Pupil performance was measured by specially designed assessments, in depth case studies and classroom observations as well as by the analysis of data from teachers; all of this was set in the context of the specific IT provision in the classes concerned. Here there was careful and consistent monitoring of the details.

THE REPORT'S FINDINGS

There is an ample and absorbing description of the work and the data that emerged in the middle section of the report (chapters 2 to 6). After the study was completed, it became possible to answer the questions that had originally been set in three main areas:

Did IT make a contribution to pupils' learning?
What of planning for teaching to include IT?
What were the demands of IT on schools?

In order to come to grips with what is in fact a potentially confusing mass of statistical data, which was handled in what to non-specialists may be intimidating ways, let us look at the global findings of each of these key findings in turn.

IT AND PUPILS' LEARNING

Yes, using IT did make an impact on the learning of the pupils studied; but this impact was far from consistent across subjects or ages. It appears to have made a greater contribution to work in mathematics and geography (particularly at secondary level) and in English (at Primary level).

One conclusion that the researchers drew was that some of the impact of IT was attributable to its motivating power. Other important factors were the successes derived from well-directed work in groups and use of appropriate methods of teaching.

Significantly, the report highlights the vital part in furthering learning that is played by the teachers' own understanding not only of the way the software works and can be used in the classroom, but also the underlying pedagogical assumptions made by it to enable the pupils to use it successfully.

This is an interesting echo of one of the points made in the last BEEBUG Education - concerning the DfE report; it has crucial implications for the degree to which teachers are adequately prepared and trained in the use of IT. To quote from the report:

> "... the role of the teacher is central and the needs for inservice and opportunities to experience or learn about new approaches and reflect on practice are paramount"
>
> (p. 159)

Again, by implication, opportunities are being missed: there is excellent software available but all too often it is under-used or misused.

PLANNING FOR USE IN THE CLASSROOM

Further, the study also found conclusively that the attitudes of teachers responsible for managing IT in their pupils' learning was important. Where teachers fully understood such factors affecting pupil performance as group organisation and collaborative work, as well as the importance of the processes of learning (as opposed to mere outcomes), then pupil progress was found to be greater. Similarly, where teachers themselves used innovative and/or imaginative teaching strategies, and showed enthusiasm for IT, then, again, progress by the pupils was measurably greater.

None of this is particularly surprising. What it does confirm, however, is that good classroom practice in a wider sense is likely to be an important factor in promoting enhanced learning when it is carried over into work with the computer.

THE DEMANDS OF IT ON SCHOOLS

The report found that 90% of schools had an IT policy on allocation and 60% had one for curriculum issues; yet by themselves - unsurprisingly enough - they were no guarantee of enhanced learning on the pupils' part.

Head teacher attitude, however, did play an important rôle. And whilst the support from LEA advisers and school IT co-ordinators was not discussed by the Kings' team, the report mentions the endorsement of the work of those holding these responsibilities that was contained in a recent study by Her Majesty's Inspectorate (HMI).

Analogously, the (lack of) in service provision to support teachers (of all backgrounds and specialisms) is seen as crucial. It is perhaps the single feature of their work most often quoted by the teachers themselves. That they perceive it as so important is something that should not be overlooked.

DIFFICULTIES

The Study found that most teachers had a number of very real difficulties in all three areas; but that those difficulties were usually associated with the management of the resource. These include: pupil access, promotion of group work, particularly in the context of the current mania for recording the minutiae of achievement often to the exclusion of truer teaching, concerns about how IT by its very nature could be fitted into the knowledge and outcome-led demands of the National Curriculum, and successful integration of IT use into (more conventional) topic planning.

The report steps back to conclude that - given a depressingly long list of barriers to effective use, awareness of the software, the pattern of use so far established in our schools may be little more than a first small step. Greater emphasis will have to be given to:

> "the wider issues of means, organisation, management and teaching style...(as the) essential ingredients for effective implementation."
>
> (p. 160)

CONCLUSIONS

Again, one is deprived of much to be optimistic about; as the experience of many pupils - in primaries in particular - is limited to infrequent exposure to the technology of ten years ago (however cherished our 8-bit BBC computer may be to us), there can be little doubt that those responsible for overall policy on the resourcing, in service provision and encouragement to use IT with discrimination are still knowingly selling a generation of school pupils very far short.

The King's ImpacT study concludes by saying that:

> "In spite of a number of commendable efforts and a sustained national strategy for the implementation of IT in education, people at all levels needed more help in formulating clear policies and strategies... (providing) a comprehensive and long term view to take full advantage of the potential impact of IT on pupils' learning"
>
> (p. 166)

As BEEBUG goes into its final half year, maybe there are readers who would like to contribute to this important debate; much of what has been said in these two short pieces in BEEBUG Education applies to the 32-bit hardware, too; please write to BEEBUG at the usual address to make your views known.

512 Forum

by Robin Burton

We're continuing with our look at back-up strategies for the 512 using PKZIP, but first we'll deal with the little teaser I left with you last month.

MULTIPLE BATCH FILES

Calling a second batch file from a first is easy; the difficulty is getting control back to the first again afterwards. In our example this meant that the second and subsequent parameters submitted to MAIN were useless; as the job stands only the first will ever work.

If you investigated you'll have found that it's not just that the parameters to MAIN are forgotten, but that control doesn't return to MAIN all. You can easily see this by adding a line at the end of MAIN such as 'ECHO Main again'; it doesn't execute because control doesn't return to MAIN after the first called file has completed.

To add interest I said any combination of subsidiary files should be capable of being run in any order, but one thing I didn't specify is that the method shouldn't be limited to three sub-files. However, clearly an ideal answer should handle any reasonable number. It can be done, it's not even difficult, and in fact there are two ways to solve the problem, although one's much more efficient than the other. Let's have a look at them.

Bear in mind that we have two objectives, depending on precisely what we're trying to do in the batch files. One is to preserve parameters in the main file through calls to subsidiary files, as in MAIN; the other can be, in effect, to pass parameters in both directions between several batch files.

The loss of control we've seen is caused by COMMAND.COM, which is pretty simple-minded. Its problem is that it can keep track of only one batch file at a time, so as soon as you call a second one, total and permanent amnesia sets in. One obvious solution therefore is to introduce extra copies of COMMAND.COM, so each can keep its own set of batch data intact. With this method the called files can remain exactly as they are; only MAIN needs alteration as follows:

```
COMMAND /C %1
COMMAND /C %2
COMMAND /C %3
```

Now all should, because the original copy of COMMAND.COM only has MAIN to handle, while the three subsidiary files are each executed within a second, separate command shell. By the way, in case you missed it a while ago, the '/C' switch tells COMMAND.COM to terminate automatically when the called program or batch file terminates. This avoids the need for an EXIT command to leave each of the second command shells.

That solves the problem of executing all three files and it also allows us to call them in any order, but it's less than ideal. The disadvantage is that every line unconditionally loads an extra copy of COMMAND.COM which not only slows down the job, but also means that COMMAND.COM must be available in a current path. Both of these can be serious minuses if you're using floppies.

Worse, each extra COMMAND.COM loads whether the relevant batch file is going to be called or not. Null parameters don't, as you might think, leave extra copies of COMMAND.COM occupying precious RAM (they just load and exit without doing anything), but they are a complete waste of time which should definitely be avoided. OK, that's easy too: put an IF EXIST test on each line, like this:

```
IF EXIST %1.BAT COMMAND /C %1
IF EXIST %2.BAT COMMAND /C %2
IF EXIST %3.BAT COMMAND /C %3
```

Now, if any of the three (or more) parameters is blank those lines won't execute, because no file called just '.BAT' will be found. Problem solved!

But that causes another. This routine will still waste a lot of time, again worst for floppies, because every line must now search the directory to see if the (parameterised) batch file exists before it can decide whether to execute or not. In fact on my winchester (my COMMAND.COM lives in the RAM disc) this version of the file takes longer than the previous one, so the fix is worse than the problem.

Time for a different approach!

A BETTER MOUSETRAP

That's the 'official' way to tackle the problem; now let's look at the best way.

This method exploits COMMAND.COM's limitations and answers all the questions with no performance implications, no matter how many parameters are supplied or omitted. For this, both the main and subsidiary files must be altered, in our example MAIN becomes a single line, like this:

```
%1 %2 %3
```

while the subsidiary files (e.g. ONE.BAT) gain a second line to become:

```
ECHO This is ONE.BAT
MAIN %1 %2
```

Note that all subsidiary files have precisely the same extra line - don't change the parameter numbers. What happens now is that MAIN calls %1.BAT (whichever that happens to be) with two parameters, %2 and %3, which are implicitly SHIFTed left one position by the call. Having loaded a second batch file the first one and its parameters are totally forgotten by COMMAND.COM. The second file executes, then calls the first again, this time with the two parameters supplied to it which are therefore passed back to the main routine unchanged.

For example, suppose initially we enter:

```
MAIN THREE ONE
```

MAIN executes its single line as:

```
THREE ONE (blank)
```

so THREE.BAT is called with one parameter, in this case 'ONE'. THREE executes, then in its second line calls MAIN again, with (now) only one parameter - 'ONE'. ONE.BAT is then called by MAIN, this time with no parameters at all. ONE.BAT executes and then calls MAIN, this time passing nothing back, so MAIN executes a blank command (which of course does nothing) and terminates. Neat eh?

Although in the example I used three subsidiary files, you can see that not only does it not matter if parameters are omitted and the order in which they're supplied is irrelevant, but the method holds for up to nine subsidiary files, In fact, with no penalty whatsoever you can include parameters %1 to %9 in the main file and %1 to %8 in all the subsidiaries, then no files will need alteration if you

find you need to add extra subsidiary routines later.

BACK TO BACK-UPS

In case you thought I'd forgotten, the point of all this, apart from providing a bit of entertainment, is that DOS batch facilities are pretty basic, and without a bit of lateral thinking they often impose limits on how you can do a job.

Taking back-ups is a perfect example of a regular and frequent job that may demand flexibility in operation, but which should remain as simple and reliable to operate as possible. I handle the differing needs for my ES back-ups (see last month) using five batch files which use exactly the above technique.

For ES back-ups MAIN.BAT is actually called SAVE.BAT, while the four subsidiary files are called DEV, NEWPROGS, SOURCES and ISSUES (BAT) each of which backs up the appropriate section of ES data. I won't list their contents here, but a few examples will let you see how they're used and perhaps you can borrow from these ideas to use for yourself.

If I've been programming, but the job isn't finished, I need to secure only ES\DEV, in which case at the end of the session I simply enter:

```
SAVE DEV
```

If I've reached the testing and debugging stage but the program still isn't completely finished, I'll obviously need to secure both ES\DEV and ES\NEWPROGS, so I enter:

```
SAVE DEV NEWPROGS
```

When I think a new program is fully tested and ready for use I copy the source file to ES\SOURCES and enter (can you guess?):

```
SAVE DEV NEWPROGS SOURCES
```

On the rare occasions that I re-secure the ES issue directories the command becomes:

```
SAVE ISSUES
```

and of course, should I ever need to secure the whole lot there's a bit more typing, but it's just as simple:

```
SAVE DEV NEWPROGS SOURCES ISSUES
```

Hopefully these examples have been 100% predictable and thoroughly boring. That's precisely how it should be. If your back-up procedures are properly designed and set-up they should be so simple that you can't possibly forget how to use them and so easy that there's no excuse for not doing so.

PKZIP DETAILS

You saw last month how straightforward a PKZIP command can be, but of course there are often one or two extras you'll need in a live situation. I've said several times that the biggest aid to quick and efficient back-ups is to avoid duplication. OK, you can avoid some files altogether such as application files, especially if they're stored separately, but that's not all you can do.

Obviously each time I secure any of the above directories only one or two files are likely to have changed or been added, so all the others can be ignored. This is where PKZIP command options, mentioned last month, are used. These allow you to modify the action of PKZIP and there are a large number of them. We'll concentrate here on the most useful for back-ups and for illustration I'll use my hard disc root directory back-up routine which, in this case, is itself a subsidiary routine (called ROOT.BAT). Here's the file:

```
echo off
:start
if exist a:root.zip goto checkb
echo INCORRECT DISC IN DRIVE A:
goto retry
:checkb
if exist b:root.zip goto secure
echo INCORRECT DISC IN DRIVE B:
:retry
ECHO Insert correct disc and try again
pause
goto start

:secure
fset a:root.zip [rw
pkzip -i- -ws a:root -x@exclude
pkunzip -t a:root
echo OK to continue? 1=Yes, 2=No
select 1 2
if errorlevel 2 exit

fset a:root.zip [ro
fset b:root.zip [rw
pkzip -i -ws b:root -x@exclude
fset b:root.zip [ro
echo on
```

As you can see I take two back-ups, so the first section of the file makes sure that I have the right discs in both floppy drives by checking for the presence of the relevant ZIP files. If they're not found the discs can be changed and the job can be re-started.

If all's well execution continues from 'secure', when first A:ROOT.ZIP is set read-write to allow updating. The next line is the first PKZIP command, so let's examine the options.

The first option is '-i-' (NB. many PKZIP and UNZIP options ARE case sensitive) which tells PKZIP to archive only those files which have the archive attribute set, but the trailing minus after the '-i' tells PKZIP not to reset the archive bit because it's needed again for the second back-up. Following that the next option, '-ws', instructs that files with the system bit set (e.g. utilities) should be included in the operation, otherwise they'd be ignored.

Next is the name of the output file which is to be updated (A:ROOT.ZIP) but with the file extension defaulted. Finally there is one further directive, '-x' which is the exclude directive. If followed by a filename the specified file is excluded from the archive operation, but if the filename is preceded by '@', as above, it instructs PKZIP to read the named file (ie. EXCLUDE) for a list of files which are to be excluded. In this case EXCLUDE contains the names DOSPLUS.SYS, 6502.SYS and COMMAND.COM. These needn't be secured because they're automatically recovered if the partition is re-created.

Next the integrity of the updated ZIP file on drive A: is checked by PKUNZIP using the '-t' (test) option, after which confirmation is needed before the second back-up proceeds. Confirmation is handled by a small program called SELECT which provides manual entry into batch files. If there were a fault, of course, the job would be abandoned, the trouble identified, and the back-up re-run.

If drive A;'s back-up file is OK, it's again set to read-only, then drive B:'s ZIP file is updated, but this time note that the trailing minus on the '-i' option is omitted, so the archive attribute of archived files is reset to prevent the same files being re-archived in future (unless they're subsequently updated again of

course). B:ROOT.ZIP is then set to read-only and the job ends.

Naturally I don't secure the root directory often, only if an existing program is revised or a new one is added. However, there's virtually nothing to remember so it doesn't matter how long it is since the last back-up, I just select the root directory, enter the command 'SAVE ROOT' and the job runs. SAVE.BAT and ROOT.BAT themselves are of course secured along with all the other files, while batch files for recovery are kept on each of the floppies (along with a working copy of PKUNZIP, think about it!)

In fact each back-up floppy usually contains several .ZIP files, so to recover the root directory I'd actually enter 'RECOVER ROOT', but you can guess how that works by now, so I won't go through it again.

The above is an example of how I secure one particular directory which happens to have some unique contents, but you can easily see how to build such an operation into a more flexible routine like the one described for the ES directories. Not all back-ups require things like file exclusion and not all directories contain system files, so many back-ups are simpler. However, other useful options we haven't yet looked at remain.

NEXT MONTH

Next month we'll complete our look at PKZIP by investigating options such as recursing sub-directories, plus of course how PKUNZIP is used when problems force recovery.

B

Mr Toad's Machine Code Corner

This month Toady looks at AND, ORA and EOR, without the help of Pooh.

Greetings, reptile readers all, and watch out for toads before you light that bonfire. First, readers' letters: Cliff Blake sent in a really excellent memory-map of all the Master's odd areas, labelled with the addresses that control them. He also seems to have solved the mystery of where those names come from. You remember: ANDY, LYNNE and that lot. I had thought they must be named for friends of the original Acorn programmers, but in a very amusing letter, Cliff assured me they must be acronyms:

FRED	Fixed Ram for External Devices
JIM	Joint Internal Memory
SHEILA	System Hardware Electronic Interface Logical Array

Now these are so plausible that for my money they're right, or very close. The trouble is, neither of us has any serious ideas about the other three. Here are Cliff's offerings:

ANDY	hANDY RAM
HAZEL	High Address Zaps Electronic Light
LYNNE	LYing oN Nearly Everything

If you think about it, those are very witty, but... no. Especially because he also pointed out that there are three C's in ACCCON. I sulked for a week. Anyway, wear your badge with pride, Cliff.

Guess what this month's competition is. Now, we still have to deal with the three bitwise operations that I didn't get round to last month: AND, OR and EOR. These are absolutely fundamental to machine code and you really must understand them if you're going to write any serious programs. Since all 6502 mnemonics have three letters, OR is called ORA in BBC assembler, but not in BBC Basic, despite which OR and ORA are in fact the same operation. Some books and other computers say XOR for EOR.

They are applied to two bytes, of which one is always in the accumulator and the other may be a number or an address. The result is always left in the accumulator. Most of the addressing modes can be used - see the manual. In each case, you can only make sense of the operation if you visualise it in binary, taking it one bit at a time. Write out the two bytes in binary, one below the other, then use the 'truth table' for that operation to write the 'answer' bit below each pair - you get a 'bottom line', just as in arithmetic.

Usually, in practice, one of the bytes operated on is unknown at the time of programming and the other, most often the one in A, is carefully worked out by you so as to have a specific effect on the unknown byte - to set, reset or 'flip' one or more of its bits. The prepared byte is often called a 'mask', perhaps because these operations are a bit like masking off an area before spray-painting; some bits are affected, others aren't. Sometimes the mask is also unknown at the time of programming but your code determines what job it will do.

AND means that the result has a given bit set (=1), only if the equivalent bit was

set in both the original numbers. The truth table is:

```
1 AND 1 = 1 : 1 AND 0 = 0 :
0 AND 1 = 0 : 0 AND 0 = 0
```

This means that ANDing a bit with 1 does nothing, it's only the clear bits of the mask which affect the outcome.

Here's the most common example of the use of AND. I'm afraid I've used it before, but it must be mentioned here: the ASCII code for capital 'A' is &41; binary 01000001. Lower case 'a' is &61; binary 01100001. The difference between the two - and between all capital letters and their lower case equivalents - is that the capitals have bit 5 clear, whereas the lower case have bit 5 set. So, if you do AND 11011111 - that's AND #&DF - on a letter which has been input into your program, you're guaranteed to end up with a capital. Note that if it was a capital to start with, it remains unchanged. As I said just now: write out lower case 'a' in binary: 01100001. Underneath it, bit for bit, write &DF: 11011111. Working from the truth table, put the result of ANDing each pair below. The bottom line is the answer. Now do the same with capital 'A'. See?

```
    01100001 (&61)
AND 11011111 (&DF)
  = 01000001 (&41)
```

Now in the bit-maps of the character set, the downstrokes are mostly two bits thick. If you aren't clear about bit-maps, sketch an eight by eight grid of boxes. Shade in any box if the bit in the following list is a zero, else leave it blank. From top to bottom, the lines go: 00111100, 00011000 five times, 00111100 and 00000000. Now stand back from the grid - you can see a capital I (with a space at the bottom).

In the Fontz ROM (BEEBUG Vol.10 No.1), I created the 'thin' font by copying each 'slice' of the bit-map of each character into workspace - we'll call it *.store*, loading the accumulator with the same byte, shifting the byte one place to the left, ANDing the result with the original byte and then putting the result back into the font:

```
LDA store
LSR A
AND store
STA font \ or that's what it boiled down
    to.
```

If you work it out, that 'thins down' 00011000 to 00010000. In this way, the ROM reduced all the two-bit-wide downstrokes, which gave a passable 'thin' font.

```
    00011000 after LSRA becomes 00110000

    00011000
AND 00110000
  = 00010000
```

ORA is the opposite of AND: if either bit of the two is set, the bit of the result is set. Truth table:

```
1 ORA 1 = 1 : 1 ORA 0 = 1 :
0 ORA 1 = 1 : 0 ORA 0 = 0
```

This means that where a bit in the mask is 0, nothing happens, but where the mask has a bit set, the other byte gets it set, too. We can use this to make every letter end up as lower case (ORA #&20), or to produce a 'bold' font with the above routine.

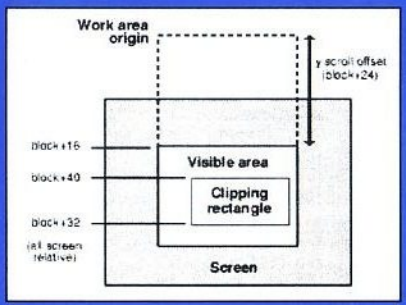
continued on page 50

RISC user

The number one subscription magazine for the Archimedes

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

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



SUBSCRIPTION DETAILS

As a member of BEEBUG you may subscribe to RISC User for the reduced rate of £18.40 (a saving of £1.50 on the normal subscription rate).

Overseas subscription rates are as follows:
£27.50 Europe and Eire, £33.50 Middle East, £36.50 Americas & Africa, £39.50 Elsewhere

Contents of the latest Vol.7 Issue 1 of RISC User:

PRO-ARTISAN 2 FROM CLARES

A review of ProArtisan 2, a new version of the popular graphics package from Clares.

GETTING THE BEST RESULTZ

A review of Resultz, the new spreadsheet and second part of the Fireworkz suite from Colton Software.

VIDEO NOUVEAU - COLOURS OF THE FUTURE

A look at what's in store video-wise for the next generation of Acorn computers, including VIDC20.

STAR SJ-144 COLOUR PRINTER

Review of the affordable new Star colour printer for the Arc.

GAMES ROUNDUP

Preview of Simon the Sorcerer from GamesWare, and a review of Stunt Racer 2000 from The Forth Dimension.

WIMP TOPICS

A major series aimed at readers interested in Wimp programs and Wimp programming. Each article looks at aspects of a particular topic.

WRITE-BACK

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

INTO THE ARC

A regular series for beginners.

TECHNICAL QUERIES

A column which answers your technical queries.

THE DOS SURVIVAL GUIDE

A series of articles on how to use the PC Emulator.

Census (continued from page 27)

```
RINTSI%(X%);:NEXT:PRINTTot%
 1640 IF BIG @%=&20105 ELSE @%=&20107
 1650 PRINT'TAB(13);:FOR X%=1TOA%(QI%):I
F Tot% PRINT 100*SI%(X%)/Tot%;"%"; ELSE
PRINT 0;"%";
 1660 NEXT:PRINT100;"%"
 1670 PROCzer:@%=&090A
 1680 *FX178,255,0
 1690 IF A=70 PRINT"Press a Key":A=GET
 1700 ENDPROC
 1710 :
 1720 DEF PROCprint
 1730  IF A=71:PRINT:VDU2,1,15:PRINT:VDU
 21:QI%=QI%(N%):QD%=QD%(N%):PROCdisplay:
VDU6,2,1,18,3:ENDPROC
 1740 PRINT:VDU2,1,15:PRINT:VDU 21:FORN%
=1TONumb%:QI%=QI%(N%):QD%=QD%(N%):PROCdi
splay:NEXT:VDU6,2,1,18,3,7
 1750 ENDPROC
 1760 :
 1770 DEF PROCzer
 1780 FORX%=1TOH%:FORY%=1TOH%
 1790 SI%(X%)=0:SD%(Y%)=0
 1800 NEXT:NEXT
 1810 ENDPROC
 1820 :
 1830 DEF PROCgetparam
 1840 ONERROROFF:C%=OPENINP$:PROCefile(C
%,P$)
 1850 INPUT#C%,Start%:INPUT#C%,End%:INPU
T#C%,Forms%:INPUT#C%,Top%:INPUT#C%,Rmax%
 1860 CLOSE#C%
 1870 ENDPROC
 1880 :
 1890 DEF PROCdate:REM LINE 1900 FOR MAS
TER ONLY
 1900 REM PRINTTAB(10);:*TIME:REM MASTER
 only
 1910 ENDPROC
 1920 :
 1930 DEF PROCefile(i%,I$)
 1940 IF i%=0 PRINTTAB(3,19)"File "+I$+"
 not found":CLOSE#i%:TIME=0:REPEATUNTIL
TIME=200:CHAIN"ANALYSE"
 1950 ENDPROC
```

```
   10 REM Program Crossdu
   20 REM Version B 1.0
   30 REM Author Paul Goldsmith
   40 REM BEEBUG November 1993
   50 REM Program Subject to Copyright
   60 :
  100 ONERROR PROCerr
  110 VDU15:VDU26
  120 PROCcode
  130 DIM I%(H%),D%(H%),Z%(5,H%,H%),T%(J
%),A%(J%),QI%(5),QD%(5)
  140 CLS:PROCt(10,"Crossdump"):PROCwind
ow1:PROCseries:PROCgetcontrol:PROCchoose
::PRINTTAB(3,20)"CROSSTABULATION COMPLET
E":TIME=0:REPEAT: UNTIL TIME =200:CHAIN"
ANALYSE"
  150 END
  160 :
 1000 DEF PROCgetcontrol
 1010 *FX200,1
 1020 ONERROROFF:CLS:C%=OPENINH$:PROCefi
le(C%,H$)
 1030 PRINTTAB(3,8);SPC(38);TAB(3,8);"Co
ntrol loading"
 1040 INPUT#C%,Head$,Q%
 1050 FORT%=1TOJ%
 1060 INPUT#C%,A%(T%):FORY%=0TOA%(T%):IN
PUT#C%,A$
 1070 NEXT:NEXT
 1080 CLOSE#C%:PRINTTAB(3,8);SPC(38);TAB
(3,8);"Control loaded":*FX200,0
 1090 TIME=0:REPEATUNTILTIME=100
 1100 ENDPROC
 1110 :
 1120 DEF PROCchoose
 1130 PRINTTAB(2,20);" AND RETURN":PROCs
elect:PROCt(10,"Analysing"):PROCwindow1:
CLS:PROCanalyse:PROCsave:VDU7
 1140 ENDPROC
 1150 :
 1160 DEF PROCanalyse
 1170 :
 1180 VDU26
 1190 PRINTTAB(6,15);"READING RECORD"
 1200 *FX200,1
 1210 C%=OPENUPD$
 1220 R%=Start%
 1230 REPEAT
```

```
 1240 PTR#C%=5*(R%-1)*Q%:B%=BGET#C%:PTR#
C%=PTR#C%-1:IF B%<>&40 GOTO 1410
 1250 FORI%=1TONum%:QX%=QI%(I%):QY%=QD%(
I%)
 1260 AI%=A%(QX%)+1:AD%=A%(QY%)+1
 1270 PRINTTAB(20,15)"    ":PRINTTAB(20,
15)R%
 1280 PTR#C%=5*(R%-1)*Q%+5*(QX%-1)
 1290 INPUT#C%,N%
 1300 FORX%=A%(QX%)TO1 STEP-1:N3%=X%-1
 1310 IF N%>=2^N3% I%(AI%-X%)=1:N%=N%-2^
N3%  ELSEI%(AI%-X%)=0
 1320 NEXT
 1330 PTR#C%=5*(R%-1)*Q%+5*(QY%-1)
 1340 INPUT#C%,N%
 1350 FORX%=A%(QY%)TO1 STEP-1:N3%=X%-1
 1360 IF N%>=2^N3% D%(AD%-X%)=1:N%=N%-2^
N3% ELSED%(AD%-X%)=0
 1370 NEXT
 1380 FORX%=1TOA%(QX%):FORY%=1TOA%(QY%)
 1390 IF I%(X%) AND D%(Y%) Z%(I%,X%,Y%)=
Z%(I%,X%,Y%)+1
 1400 NEXT:NEXT:NEXT
 1410 R%=R%+1:UNTIL R%=End%+1
 1420 CLOSE#C%:*FX200,0
 1430 REM PRINTTAB(3,22);"Time off ";:PR
OCtime
 1440 ENDPROC
 1450 DEF PROCselect
 1460 CLS:PRINTTAB(5,4)"SELECT CROSS TAB
S"
 1470 INPUTTAB(3,5)"TYPE NO OF CROSS TAB
S UP TO 5 "Num%:IF Num% >5 VDU7:GOTO 147
0 ELSE IF Num%<1 VDU7:GOTO 1470
 1480 FOR I%=1TONum%
 1490 INPUTTAB(5,I%+6)"Ind "QI%(I%)
 1500 INPUTTAB(15,I%+6)"Dep "QD%(I%)
 1510 NEXT
 1520 ENDPROC
 1530 :
 1540 DEF PROCseries
 1550 INPUTTAB(3,10)"TYPE A CHARACTER TO
 DISTINGUISH"''"SELECTION OF CROSSTAB";S$
:IF S$="" OR LEN(S$)>1 GOTO1550
 1560 PROCgetparam
 1570 ENDPROC
 1580 :
 1590 DEF PROCsave
 1600 E$=D$+S$
 1610 *FX200,1
 1620 C%=OPENOUT E$:IF C%<1 Rep$="File "
+E$+" not found":PRINTTAB(1,8);Rep$:Rep$
="":GOTO 150
 1630 PRINT#C%,Num%:FOR I%=1TO Num%:PRIN
T#C%,QI%(I%),QD%(I%)
 1640 FORY%=1TOH%
 1650 FORX%=1TOH%
 1660 PRINT#C%,Z%(I%,X%,Y%)
 1670 NEXT:NEXT:NEXT
 1680 CLOSE#C%:*FX200,0
 1690 ENDPROC
 1700 :
 1710 DEF PROCcode
 1720 ONERROROFF:C%=OPENIN"CODE":PROCefi
le(C%,"CODE")
 1730 INPUT#C%,F$,J%,H%,K%:CLOSE#C%:H$=F
$+"HED":D$=F$+"DA":P$=F$+"PAR"
 1740 ENDPROC
 1750 :
 1760 DEF PROCtime:REM MASTER ONLY
 1770 *TIME
 1780 ENDPROC
 1790 :
 1800 DEF  PROCgetparam
 1810 ONERROROFF:C%=OPENINP$:PROCefile(C
%,P$)
 1820 INPUT#C%,Start%:INPUT#C%,End%:INPU
T#C%,Num%
 1830 CLOSE#C%
 1840 ENDPROC
 1850 :
 1860 DEF PROCerr
 1870 REPORT:PRINT" at line "ERL
 1880 PRINT"Press a key ":A=GET:PRINT"(S
)top (Q)uit (R)un":A=GET
 1890  IF A=83 END ELSE IF A=81 CHAIN"AN
ALYSE" ELSE IF A=82 RUN
 1900 ENDPROC
 1910 :
 1920 DEF  PROCefile(i%,I$)
 1930 IF i%=0 PRINTTAB(3,19)"File "+I$+"
 not found":CLOSE#i%:TIME=0:REPEATUNTIL
TIME=200:CHAIN"ANALYSE"
 1940 ENDPROC
```

```
 1800 IF NOT(DC%) OSCLI("SRWRITE "+STR$~
(s%)+" "+STR$~(f%)+" "+STR$~(p%)):ENDPRO
C
 1810 PTR#ch%=p%:FOR A%=s% TO f%-1:IF r%
 ?A%=BGET#ch%:ELSE BPUT#ch%,?A%
 1820 NEXT:ENDPROC
 1830 :
 1840 DEF PROCsee(I%):LOCAL I$:I$=STR$(I
%):IF I%=0 I$="New"
 1850 PROCclear:PROCtitle(N$,1):PROCtitl
e("M-Base Item :"+CHR$(131)+I$,22)
 1860 IF I%<>0 PROCread(I%):ELSE FOR A%=
1 TO F%:T$(A%)="":NEXT
 1870 FOR A%=1 TO F%:PRINTTAB(X%(A%),Y%(
A%));F$(A%);T$(A%):NEXT:ENDPROC
 1880 :
 1890 DEF PROCalter(J%)
 1900 PROCsee(J%):v%=1:REPEAT
 1910 IF NOT PR% T$(v%)=FNinput(F$(v%),T
$(v%),X%(v%),Y%(v%),L%(v%),32,126)
 1920 IF INKEY-58 AND v%>1 v%=v%-1
 1930 IF INKEY-42 v%=v%+1
 1940 UNTIL v%>F% OR INKEY-122 OR PR% OR
 (key%>0 AND key%<12)
 1950 IF PR% PROCprint:ENDPROC
 1960 PROCwrite(J%):ENDPROC
 1970 :
 1980 REM ***** Main menu *****
 1990 :
 2000 DEF PROCmenu:VDU23;11,0;0;0;0;0;
 2010 T$="M-Base :"+CHR$(131):IF N$<>""
T$=T$+N$:ELSE T$=T$+"No File"
 2020 CLS:PROCtitle(T$,1)
 2030 PRINTTAB(0,5);"Options :"''" 1) Vi
ew records"'" 2) Add records"'" 3) Delet
e records"'" 4) New or alter database"'"
 5) View information"
 2040 IF DC% PRINT" 6) Open file"'" 7) C
lose file":ELSE PRINT" 6) Load file"'" 7
) Save file"
 2050 PRINT" 8) Search for records"'" 9)
 Status"'" 0) Exit"
 2060 PRINT''"Select 1-0 :";CHR$(130);
 2070 REPEAT O$=GET$:O%=VAL(O$):UNTIL O%
>0 OR O$="0" OR O$="*" OR O$="+":IF O$<>
"*" AND O$<>"+" PRINTO$:ELSE O%=9+INSTR(
"*+",O$)
 2080 ENDPROC
 2090 :
 2100 DEF PROCoptions
 2110 IF O%=0 ENDPROC
 2120 IF N%=0 AND INSTR("138",STR$(O%))
PRINT'CHR$136;CHR$131;"No records":VDU7:
TIME=0:REPEATUNTILTIME>=150:PROCclear:EN
DPROC
 2130 ON O% PROCview , PROCadd , PROCdel
ete , PROCnewalter , PROCinfo , PROCload
 , PROCsave , PROCsearch , PROCstatus ,
PROCoscli , PROCcomm
 2140 ENDPROC
 2150 :
 2160 REM ***** New/Alter Database *****
 2170 :
 2180 DEF PROCnewalter
 2190 PROCclear:PROCtitle("New / Alter d
atabase",1)
 2200 PRINTTAB(0,5);"  1) New database"'
'"  2) Add fields"''"  3) Alter screen"'
'"  4) Grab old database"''"  5) Main me
nu"
 2210 PRINT''''"Press 1-5 : ";:REPEAT:K$=
GET$:UNTIL INSTR("12345",K$):IF K$="5" E
NDPROC
 2220 IF K$="1" PROCnew:ENDPROC
 2230 IF K$="2" PROCaddfields:ENDPROC
 2240 IF K$="4" AND DC% VDU7:PRINT'"You
can't : Disc media selected":TIME=0:REPE
AT:UNTIL TIME>=700:ENDPROC
 2250 IF K$="4" PRINT'"Are you sure ?":k
$=GET$:IF k$<>"Y" AND k$<>"y" ENDPROC
 2260 IF K$="4" PROCassert:ENDPROC
 2270 PROCclear:PROCpage:ENDPROC
 2280 :
 2290 DEF PROCnew:PROCclear
 2300 PROCtitle("New file",1)
 2310 PROCtitle("M-Base",22)
 2320 N$=FNinput("Name :"+CHR$(130),"",0
,4,10,48,122)
 2330 IF N$="" ENDPROC
 2340 L%=4:N%=0:F%=0:E%=0:PROCfields:IF
DC% ch%=OPENOUT N$:CLOSE#ch%:ch%=OPENUP
N$
 2350 ENDPROC
 2360 :
 2370 DEF PROCfields
 2380 PROCcentre("Fields",14):PRINT'"Ple
```

```
ase enter field names and max. length"''"
Blank title  : Same as last title"''"Bla
nk length : No more fields":VDU28,0,12,3
9,6
 2390 REPEAT:CLS:F%=F%+1:PRINT"Field :";
CHR$(131);F%
 2400 F$(F%)=FNinput("Title  :"+CHR$(134
),"",0,2,15,32,126)
 2410 L%(F%)=VAL(FNinput("Length :"+CHR$
(129),"",0,4,3,48,57)):L%=(L%+L%(F%)+1)
 2420 UNTIL L%(F%)=0 OR F%=24:IF L%(F%)=
0 F%=F%-1
 2430 PROCclear:PROCpage:ENDPROC
 2440 :
 2450 DEF PROCpage:VDU23;11,0;0;0;0;0;
 2460 PROCtitle(N$,1):PROCtitle("Positio
n Fields",22)
 2470 LOCAL X%,Y%,A%:FOR A%=1 TO F%:IF F
$(A%)="" F$(A%)=F$(A%-1)
 2480 X%=0:Y%=4:PRINTTAB(0,4);F$(A%)
 2490 REPEAT:OX%=X%:OY%=Y%
 2500 IF INKEY-98 AND X%>0 X%=X%-1
 2510 IF INKEY-67 AND X%<(39-LEN(F$(A%))
) X%=X%+1
 2520 IF INKEY-73 AND Y%>3 Y%=Y%-1
 2530 IF INKEY-105 AND Y%<21 Y%=Y%+1
 2540 IF OX%<>X% OR OY%<>Y% PRINTTAB(OX%
,OY%);SPC(LEN(F$(A%)));TAB(X%,Y%);F$(A%)
:IF A%>1 FOR B%=1 TO A%-1:PRINTTAB(X%(B%
),Y%(B%));F$(B%);SPC(L%(B%));CHR$255:NEX
T
 2550 UNTIL INKEY-99:X%(A%)=X%:Y%(A%)=Y%
:REPEAT:UNTIL NOT INKEY-99:NEXT:PROCclea
r:ENDPROC
 2560 :
 2570 DEF PROCaddfields
 2580 LOCAL OL%,OF%,NF%,NL%,j%,h%
 2590 PROCclear:OF%=F%:OL%=L%:PROCtitle(
"Add New Fields",1):PROCfields:NL%=L%:NF
%=F%
 2600 IF OF%=F% ENDPROC
 2610 IF N%=0 ENDPROC
 2620 PROCtitle("Please wait",22):PROCti
tle("Updating database records",1):PRINT
TAB(0,5);" Updating the database to acco
madate    the new fields. May take some
time."
 2630 FOR j%=N% TO 1 STEP-1
 2640 L%=OL%:F%=OF%:PROCread(j%)
 2650 FOR h%=OF%+1 TO NF%:T$(h%)="":NEXT
 2660 L%=NL%:F%=NF%:PROCwrite(j%)
 2670 PRINTTAB(0,9);"Done ";INT((N%-j%+1
)/N%*100);"%"
 2680 NEXT:ENDPROC
 2690 :
 2700 REM ***** View Records *****
 2710 :
 2720 DEF PROCviews(s%)
 2730 v%=1:p%=s%:PROCsee(p%):REPEAT
 2740 IF NOT PR% T$(v%)=FNinput(F$(v%),T
$(v%),X%(v%),Y%(v%),L%(v%),32,126)
 2750 IF PR% PROCprint
 2760 IF INKEY-58 AND v%>1 v%=v%-1
 2770 IF INKEY-42 AND v%<F% v%=v%+1
 2780 IF PR% AND n%>0 p%=n%:PROCsee(n%)
 2790 IF (INKEY-26 OR key%=136 OR key%=2
) AND l%>0 PROCwrite(p%):p%=l%:PROCsee(l
%)
 2800 IF (INKEY-122 OR key%=137 OR key%=
1) AND n%>0 PROCwrite(p%):p%=n%:PROCsee(
n%)
 2810 UNTIL INKEY-99 OR (PR% AND n%=0) O
R  key%=3 OR key%=4:old%=p%:IF PR% PROCp
rint
 2820 IF key%<>3 AND key%<>4 PROCclear
 2830 ENDPROC
 2840 :
 2850 DEF PROCview
 2860 LOCAL q%:q%=S%:IF PR% PROCviews(S%
):ENDPROC
 2870 REPEAT:PROCviews(q%)
 2880 PRINTTAB(0,22);SPC(80);:IF key%<>4
 S$=FNinput("String : ","",0,22,25,32,12
6):PRINTTAB(0,22);SPC(40):q%=0:S$=FNlow(
S$):ELSE q%=old%
 2890 PROCtitle("Searching",22)
 2900 REPEAT:q%=q%+1:PROCread(q%):T$=FNl
ow(T$(1)):UNTIL INSTR(T$,S$) OR q%=N%:IF
 INSTR(T$,S$)=0 q%=old%:PROCtitle("Not f
ound",22):TIME=0:REPEAT:UNTIL TIME>250
 2910 UNTIL S$="":ENDPROC
 2920 :
 2930 DEF PROCprint
 2940 VDU2,21:FOR A%=1 TO F%
 2950 IF P%(A%) PRINTT$(A%);STRING$(L%(A
%)-LEN(T$(A%))+2," ");:IF CR% PRINT
```

```
 2960 NEXT:PRINT:VDU3,6,3
 2970 ENDPROC
 2980 :
 2990 REM ***** Add Records *****
 3000 :
 3010 DEF PROCadd:REPEAT:PROCadd2:UNTIL
FALSE
 3020 :
 3030 DEF PROCadd2:n%=0:l%=0
 3040 PROCsee(0):v%=1:REPEAT
 3050 T$(v%)=FNinput(F$(v%),T$(v%),X%(v%
),Y%(v%),L%(v%),32,126)
 3060 IF INKEY-58 AND v%>1 v%=v%-1:ELSE
v%=v%+1
 3070 UNTIL v%>F% OR INKEY-99
 3080 IF N%=0 S%=1:N%=1:n%=0:l%=0:PROCwr
ite(1):PROCclear:ENDPROC
 3090 N%=N%+1:PROCwrite(N%):T$(0)=T$(1)
 3100 n%=S%:REPEAT:c%=n%:PROCread(n%):UN
TIL T$(1)>T$(0) OR n%=0
 3110 IF n%=0 AND T$(0)>=T$(1) n%=N%:PRO
Cwrite(c%):PROCread(N%):l%=c%:n%=0:PROCw
rite(N%):PROCclear:ENDPROC
 3120 IF c%=S% l%=N%:S%=N%:PROCwrite(c%)
:PROCread(N%):n%=c%:l%=0:PROCwrite(N%):P
ROCclear:ENDPROC
 3130 k%=l%:l%=N%:PROCwrite(c%)
 3140 PROCread(k%):n%=N%:PROCwrite(k%)
 3150 PROCread(N%):n%=c%:l%=k%:PROCwrite
(N%):PROCclear:ENDPROC
 3160 :
 3170 REM ***** More To Follow *****
 3180 :
 3190 DEF PROCdelete
 3310 DEF PROCsave
 3450 DEF PROCload
 3570 DEF PROCoscli
 3720 DEF PROCinfo
 3830 DEF PROCstatus
 4440 DEF PROCcomm
 4760 DEF PROCsearch
 4770 PRINT'CHR$(131);"Not implemented y
et!":TIME=0:REPEAT:UNTIL TIME>=500:PROCc
lear:ENDPROC
```

Mr Toad's Machine Code Corner (continued from page 44)

EOR (Exclusive OR), gives a 1 if *only one* of the two original bits is set. If both are 1 or both are 0, you get 0. Truth table:

```
1 EOR 1 = 0 : 1 EOR 0 = 1 :
0 EOR 1 = 1 : 0 EOR 0 = 0
```

This means that if you EOR a bit with 0, nothing is changed, but where a bit of the mask is set (=1) you 'flip' or 'toggle' the corresponding bit of the other byte from 1 to 0 or 0 to 1. EOR a byte twice with the same number and you get back to the original. This fact has been used in dozens of encoding programs, (including one of mine) in BEEBUG and elsewhere. EOR a letter with a 'key', and you've got it in code. Do it again, and it's back in clear: one routine does both jobs.

There are lots of other applications for all these operations; I've only been able to give a few obvious examples here. If you keep them in mind, you'll think of them when there's a job which they can do in your program: don't be one of those programmers who permanently work with a 'Reduced Instruction Set' because there are some operations which they've never used and so never think of. Just say this little mnemonic: AND #0 clears, ORA #1 sets, EOR #1 flips.

Exercise 1: write out from memory the entire 6502 instruction set. Check with the book. If you missed more than three, make a big badge announcing: "I'M A RISP" and wear it for a week.

Next month: American scientists find a 250 million year old RAM chip, and by RMA cloning they recreate... The XY Rainbow? The Amstrog with the funny little discs? British Rail's C64? Tune in and find out.

Sideways Scrolling

The Beeb has a ready-made command to perform a sideways scroll:

```
VDU 23;13,A;0;0;
```

where A represents the new position on the screen. To see it in operation, try the following short program.

```
10 FOR A=1 TO 20
20 TIME=0:REPEAT UNTIL TIME=100
30 VDU 23;13,A;0;0;
40 NEXT
```

To reverse the direction of the scroll alter line 10 to:

```
10 FOR A=20 TO 1 STEP -1
```

To change the scroll speed, alter the loop value in line 10; for example:

```
10 FOR A=1 TO 100
```

to slow the scroll down, and:

```
FOR A=1 TO 10
```

to increase the speed.

Musical Keyboards

J Fenton

The following single line program will turn your keyboard into a piano, with the QWERTY and ZXCVBN rows playing the white notes, and the respective rows above these playing the black notes.

```
CLS:REPEAT:G$=GET$:PRINT G$:SOUND&11, -15,21+4*INSTR("Q2W3ER5T6Y7UI9OOPZSXDCFVBHNJM,L.;/",G$),10:UNTIL G$="1":SOUND&11,0,0,0
```

Debugging with EVAL

G Weston

The following short procedure may prove very useful when debugging Basic programs. Simply insert the line PROCtest at the trouble spot in your program, and then enter the variable name that you want to examine.

```
2000 DEF PROCtest
2010 LOCAL Z$
2020 INPUT "Var."Z$
2030 IF Z$<>"99" PRINT EVAL(Z$):PROCtest
2040 ENDPROC
```

Type 99 to return control to your program.

Listing Programs in Page Mode

P LeSueur and J Jakubovics

When a program is listed with Ctrl-N selected, it is very easy to forget to disengage page mode with Ctrl-O afterwards. This can prove extremely irritating when Shift has to be pressed to make a program function correctly. The solution is to define a function key to select page mode, do the listing, and then disengage page mode once the listing is finished. The only problem that can arise with this is that if Escape is pressed while the program is being listed, the keyboard buffer is cleared, and the Ctrl-O is never issued. Thus the following definition also includes a *FX230,1 command to stop the buffer being flushed, and a *FX230,0 to reset it to its default.

```
*KEY 0 *FX230,1|M|NL.|M|O*FX230,0|M
```

Deleting Lines

D Fletcher

When deleting lines from the end of a program, it is quickest to delete all but the last line in one go, and only then delete the last line.

Bird Sounds

I Evans

Here is a short program to generate bird-like sounds. It can of course be modified to taste, although the duration parameter in the SOUND statement has to be 3 for best effect.

```
10 ENVELOPE 1,1,5,5,-10,30,30,30,50,0,0,100,100
20 FOR I=1 TO 100
30 SOUND 1,1,130+RND(40),3
40 NEXT
```

B

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.

WANTED: COMAL language for the Master 128. Tel. Sheffield 0742 367611.

Master 128 with 3.5" and 5.25" drives £175, Canon PW1080A 9 pin NLQ printer £60, various manuals, books, Viewstore, Pascal etc. also available. Tel. Blackpool 0253 892709 after 5pm.

Watford dual 5.25" 40/80T switchable disc drive £30, Morley teletext adaptor with ATS, 2.59, ROM £40, Printmaster, Wordwise Plus 2, Spellmaster, Spellcheck III and Master ROMs for 128 all with appropriate manuals £75 or will split, Wordease, Master cartridges and assorted software also available. Tel. Hemel Hempstead 0442 259054.

BBC B+ (inc. Interword/Viewsheet), disc drive (Watford), Acorn RGB colour monitor, Masterfile 2, sold separately for sensible offers. Tel. Luton 0582 882502.

WANTED: Synergy Software's Sharemaster or Sharemaster Plus together with manual for use on a BBC Master. Tel. Cheshire 0270 625691.

Master 512 + turbo 2nd processor, 512 board with enhancement to 1Mb housed in Watford co-processor adaptor, high definition colour monitor and mouse, resident ROMs include View, Viewsheet, Master ROM, Wordwise+, Spell Master, also Akther 40/80 double disc drive in monitor stand with PSU, all the four Gem discs with Gempaint and Gemwrite, all well looked after and in perfect condition, all manuals £350 o.n.o. Many more ROMs, disc software and books for sale, owner updating to more sophisticated computer. Tel. Wembley 081-902 8612.

BBC B with shadow RAM/ROM expansion board, 5.25" single D/S drive with DDFS, Wordwise+, Spellmaster, continuous processing ROM, Epson FX80 printer, all in good working order and with leads and manuals £200, Watford ROM/RAM board and Spellmaster ROM, both new, unused, boxed £25 each. Tel. Gwynedd 0341 422006.

BBC model B, green screen monitor, double drives (800k), interfaced daisywheel, wordprocessor, spreadsheet, database etc. £250 o.n.o. Tel. 0392 434149.

BBC Master 512, Phillips CM8833 colour monitor, Watford twin 40/80T 5.25" disc drives, Watford utilities disc, Watford Quad ROM cartridge, Dumpout 3 ROM, Essential Software co-pro memory upgrade, Shibumi Soft Problem Solver, T.C.S. mouse driver and utilities disc 1, Dabs Press Master 512 User Guide and disc, good condition, original boxes - buyer collects or pay carriage, also BEEBUG magazines Vol.8 No.1 to Vol.12 No.4 £500 o.n.o. Tel. Somerset 0458 43906.

BBC B with DFS + ATPL ROM board, mono monitor Watford Video Digitiser, AMX mouse and modem, software, manuals. Offers? Tel. Bracknell 0344 426878.

WANTED: 512 board for BBC Master 128 (preferably expanded to 1Mb), also 65c102 turbo board and sideways cartridge. Tel. 031-650 2367.

B/W 12" monitor, Citizen 120-D 9 pin printer (two) with spare ribbons, disc drives 5.25" and 3.5", all with appropriate leads etc. reasonable offers for each accepted. Please write to David, 1 Highfield Court, St Margarets Road, London E12 5DW.

HELP WANTED: I have a BBC B with Kenda DMFS with PADS and want sideways RAM and ROM boards. Please write to David, 1 Highfield Court, St Margarets Road, London E12 5DW.

BBC B ADFS, 32k shadow RAM, 16k RAM/ROM + ZIF £50, Acorn teletext adaptor ATFS £20, Epson FX80 £40, Nidd Valley mouse, Illustrator, Chauffeur £10, CC Interbase, Mega 3 ROM, Spellmaster £20 each, Printmaster, NLQ ROM, Wordwise+, Printwise, Publisher £10 each, Toolkit Plus, disc utilities, graphics ROMs, Z88 link £8 each, Elite, Revs, Aviator + lots of games and utilities on disc £5 or under. All manuals for above, BEEBUG complete and books, offers accepted. Tel. Bath 0225 330769.

A3000 4Mb RISC OS 3.1, AKF17 monitor £450, SCSI 40Mb hard drive and interface £150, 2Mb upgrade £25, 5.25" drive and interface, DFS reader £75, Canon PW1080 NLQ printer £40. Tel. Tadcaster 0937 835744.

BBC Micro ROM book (Smith) £4, Programming the 6502 (Zaks) £5, Basci Programming on the BBC Microcomputer (Cryer) £2 Assembly Language Programming on the BBC Micro (2 books - Birnbaum, Ferguson & Shaw) £4 each, Fleet Street Editor (discs & manual) £10, or £25 the lot. Tel. Middlesex 0923 824063.

Multitasking link Archimedes - BBC (Ivoryash) disc and lead £25. Tel. 0252 710219.

Zenon, orbital, Clogger, P.I.A.S 1,6,8,9, Pipeline, Predator, Elite, Exile, Barbarian, Barbarian II, Time & Magik, Gnome Ranger, Lancerlot, The Last Ninja £50, the lot or offers? Buyer collects. Tel. Dumfries 0387 54526 ask for Colin.

BEEBUG magazines volumes 1 to 12 all bound £30, Advanced Disc User Guide £8, 30hr Basic £2, Advanced User Guide £2, BBC Disc System User Guide (temporary photocopy) £1. Tel. Hexham 0434 606684.

15" RGB colour monitor, twin 40/80 Cumana disc drives, Amstrad DMP1 9 dot matrix (tractor), Plug-in ROMs, Viewsheet, Viewstore Database, View Printer Driver, MiniOffice II, Wapping Editor DTP, Advanced Disc Toolkit, NLQ Font Designer, Penpal 2 (lightpen), Quest-paint, Quest mouse, Pace Nightingale Modem, teletext adaptor, user port splitter (all manuals for above), 30 Micro User discs £250. Tel. Surrey 081-337 4247.

Micro User magazine discs 80T 5.25" 40 original discs from May 1989 to September 1992 £1 each plus postage. Tel. Dorset 0305 852276.

BBC B ATPL ROM/RAM card 16k RAM £10, Wordwise Plus £10, boxed, manuals. Tel. 0925 268984 eves.

BBC B, View, Watford Electronics ROM/RAM board, manuals, user guide, 20 Acorn Users £55 o.n.o. 12" green screen monitor £35 o.n.o. Also Brother EP44 mains or battery portable typewriter/printer, suitable for use with Beeb. Manuals, cables and box of paper £55 o.n.o. Tel. Chesterfield 0246 810978.

Econet SJ Fileserver with 20Mb hard disc, SJ tapestreamer and several 35Mb tapes, preferably to be sold together. Tel. Bath 0225 464313.

BBC B, including sideways ROM/RAM board with Wordwise Commstar, AMX mouse and other ROM software, twin 40T DS 5.25" disc drive, Microvitec CUB colour monitor, MiniOffice II, Scrabble, XOR and other disc and tape software, plus books and manuals £250. Also complete set of bound BEEBUG magazines volumes 1 to 6 and Acorn User magazine complete from March 1983 to June 1990, no reasonable offers refused. Tel. Wales 0437 765441 eves.

BBC Master complete with mother board £80, econet inc. ROM £20, PSU £25 o.n.o. Can be seen working. Tel. Bath 0225 464161.

WANTED: Video Digitiser and also a 3.5" 80T disc drive all for the BBC Master. Tel. Bath 0225 464161.

WORDWISE-MAIL

In the most recent issue of BEEBUG (Vol.12 No.5) I read with considerable interest the review of Wordwise-Mail, a new package to help Wordwise Plus users.

The software does appear to have many restrictions in its use, but has a modest price of slightly under £10. There is no doubt that Chris Robbins' comments in the review will be of considerable help to the writer.

Readers may be reminded that there is an excellent package, Corplan, which has been with us since 1985, and has been under continuous development. It offers the user, with Wordwise Plus and Wordwise Plus II, a full range of facilities for handling correspondence, reports, invoices and many other types of 'written' work. Corplan also supports the use of a repertoire of 'forms' each defining the layout of a particular type of document for the user.

I have been using Corplan for a long while, and would highly recommend this package to my fellow members. It has saved me a tremendous amount of time in my work.

John Battersby

AND MORE ON WORDWISE-MAIL

I am surprised to see that you have devoted space in the October issue of BEEBUG to a review of Wordwise-Mail when there is already available a proven package which has none of the restrictions and drawbacks that you have found endemic in the program under review.

Following your review of Corplan several years ago, I purchased this package and subsequently was able, with valuable support from Colin Robertson the author, to incorporate almost all my computer operations within Corplan. These are connected with my recordings for conferences, including letter writing, invoicing, compiling and printing of programmes, order forms, cassette and packet labels, etc, etc.

Previously I had some 20 floppy dics in use for my operations, but since adopting Corplan, I have been able to reduce this to three, with back-ups of course. Based on several years experience, I consider Corplan at around £17 to be much better value for money than Wordwise-Mail

Harry Ryder

While Wordwise-Mail may have some limitations when compared with Corplan, it is only by publishing a full review that readers can be properly informed, and the author feel that his product has been fairly treated.

AND NOW FOR CORPLAN

In Vol.12 No.5 you published a review of Wordwise-Mail. It is unfortunate that this product, although cheap, was found to have so many restrictions in its use. However, I expect some improvements will be made eventually.

Meanwhile, readers may like to know that Corplan (of which I am the author) which you reviewed in BEEBUG Vol.10 No.8 is still available from me in its latest (1993) version at the reduced price of £17.50. Corplan is a full-featured correspondence management system of proven effectiveness and reliability that works happily with all versions of Wordwise Plus and Wordwise Plus II. It comes with full and accurate printed documentation and unlimited after-sales support.

Colin Robertson, Three Gables, 7A Talbots Drive, Maidenhead, Berks SL6 4LZ, tel. 0682 24591.

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.

RENEWAL RATES FOR BEEBUG MAGAZINE AND MAGAZINE DISC SUBSCRIPTIONS

See May 1993 Editorial for further explanation

The table below shows the renewal rate applying after the June issue 1993 according to the first issue of the renewal period. For joint BEEBUG/RISC User subscriptions add half the appropriate BEEBUG renewal rate to the full RISC User renewal rate; (UK £18.40, Europe & Eire £27.50, Middle East £33.50, Americas & Africa £36.50, Elsewhere £39.50).

Renewal Issue	Issues to go	Mag UK	Mag Europe	Mag Mid-E	Mag Am+Af	Mag Else	Disc UK	Disc O'Seas
Jun	9	16.56	24.75	30.15	32.85	35.55	45.00	50.40
Jul	8	14.72	22.00	26.80	29.20	31.60	40.00	44.80
A/S	7	12.88	19.25	23.45	25.55	27.65	35.00	39.20
Oct	6	11.04	16.50	20.10	21.90	23.70	30.00	33.60
Nov	5	9.20	13.75	16.75	18.25	19.75	25.00	28.00
Dec	4	7.36	11.00	13.40	14.60	15.80	20.00	22.40
J/F '94	3	5.52	8.25	10.05	10.95	11.85	15.00	16.80
Mar '94	2	3.68	5.50	6.70	7.30	7.90	10.00	11.20
Apr '94	1	1.84	2.75	3.35	3.65	3.95	5.00	5.60

BACK ISSUE PRICES (per issue)

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

Magazines and discs are postcode a
Please add the cost of p&p when ordering.
When ordering several items use the highest price code, plus half the price of each subsequent code. UK maximum £8.

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

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 840303, FAX: (0727) 860263
Office hours: 9am-5pm Mon-Fri Showroom hours: 9am-5pm Monday to Saturday
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by RISC Developments Ltd.

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

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 in machine readable form using plain text format if possible for text, but please ensure an adequate written description is also included of your submission and the contents/format of your disc.

In all communication, please quote your membership number.

Printed by Arlon Printers (0923) 268328 ISSN - 0263 - 7561

Magazine Disc

NOVEMBER 1993

M-BASE - We present this month the first part of a comprehensive and generalised database system for the Beeb.

SAVINGS - Use this program to explore how savings invested for the future will dwindle away over the years.

MANIC MECHANIC - This is another classic game from the BEEBUG archives. Manic Mechanic is a first rate platform style game with excellent graphics, and challenging game play.

BEEBUG WORKSHOP - The disc includes the complete set of routines for implementing virtual arrays and direct file access as described in the magazine.

MISSILE NAVIGATION - Explore the problems of this interesting subject with our expert Cliff Blake.

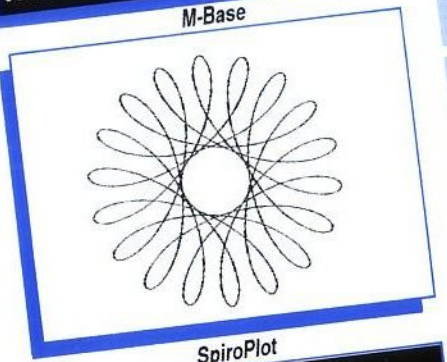
WORDWISE USER'S NOTEBOOK - The disc contains all the files described in the magazine, including Basic program, Wordwise segment program and other files.

SPIRO - This is a short implementation of the well-known spirograph type of display providing hours of fascination with an endless variety of results.

CENSUS PT4 - A complete collection of all the Census programs including the cross-reference programs from this concluding part.

MAGSCAN DATA - Bibliography for this issue of BEEBUG (Vol.12 No.6).

M-Base



SpiroPlot

Financial Futures

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.

FOR DISC (5.25" or 3.5") RENEWAL SUBSCRIPTION RATES (UK AND OVERSEAS) PLEASE SEE TABLE ON FACING PAGE

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

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

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

BEEBUG

November 1993

Special Offer

A5000 Computers for only £850

(Prior to Sept'93 the same system was £1285)

Once again we have managed to obtain a number of A5000 Learning Curve systems at a ridiculously low price. On the previous occasion we sold all 100 in only **ten days!** So hurry if you are not to miss out this time.

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

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

A5000 Specification

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

Learning Curve pack

PC Emulator v1.8
DOS 5
First Word Plus
Acorn DTP
Audio Training Tape
Pacmania
Genesis Plus

Monitor options:

Standard (AKF40)	£149
MultiScan (AKF18)	£179

RAM Upgrades:

2-4Mb	£85
2-8Mb	£379

Additional Hard Drives:

160Mb	£199
260Mb	£279
450Mb	£449

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

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