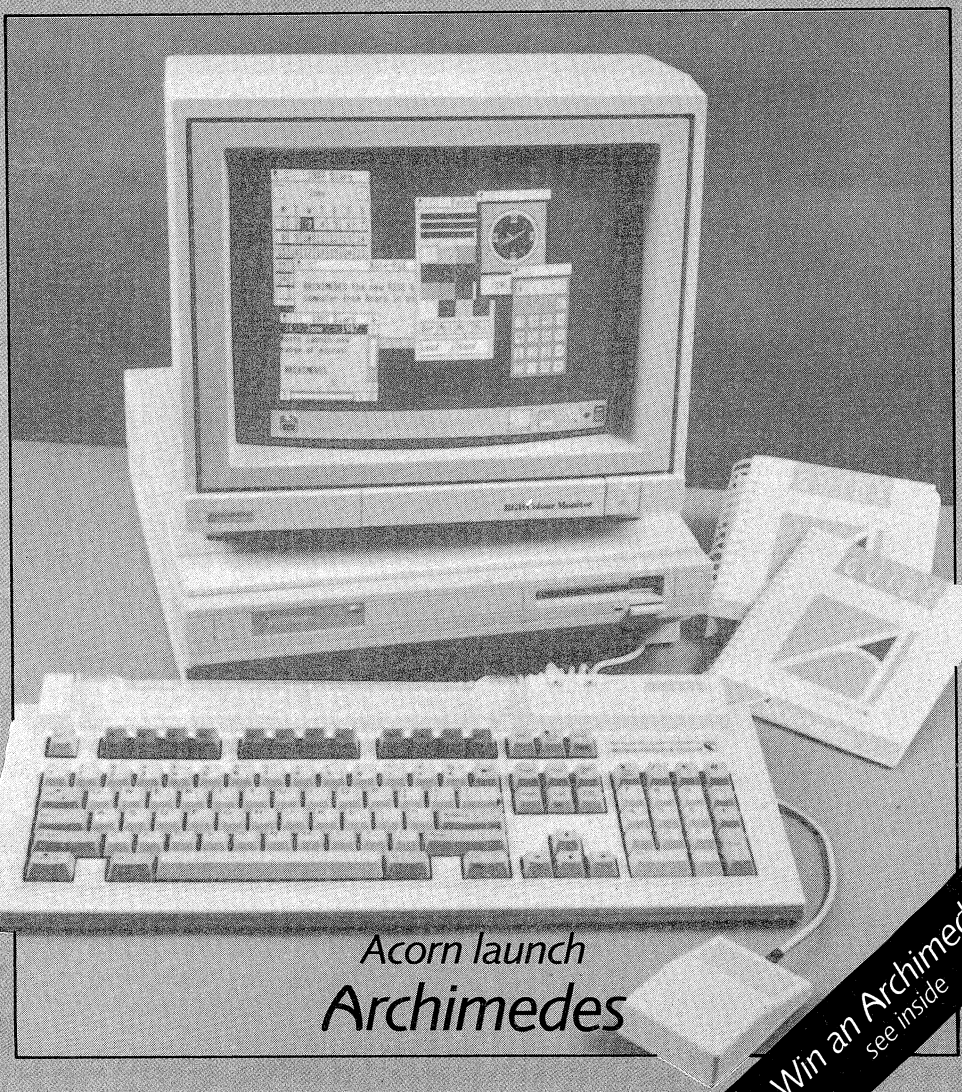
VOLUME 6 NUMBER 3

BEEBUG

JULY 1987

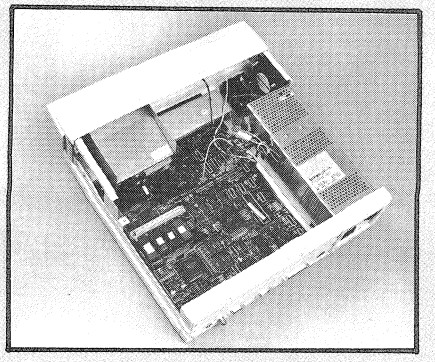
PERSONAL ADDRESS BOOK

FOR THE BBC MICRO AND MASTER SERIES

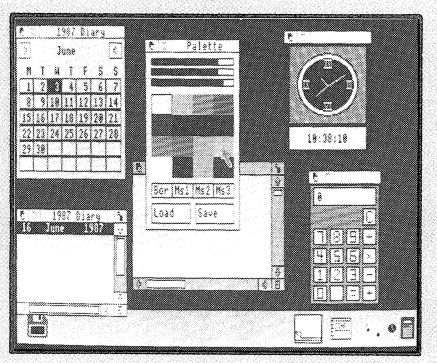


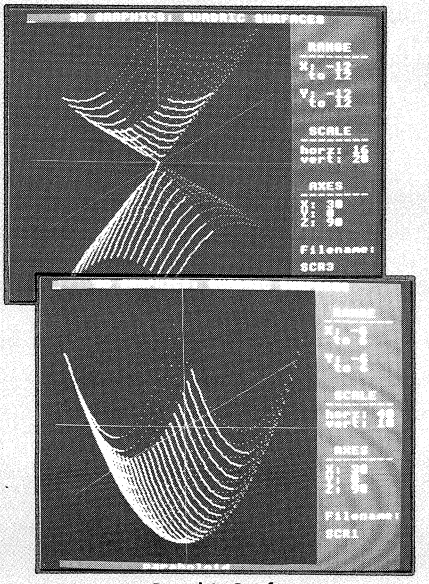
Acorn launch
Archimedes

Win an Archimedes
see inside



Archimedes





Quadric Surfaces

BEEBUG

Volume 6 Number 3 July 1987

FEATURES

Acorn Launch Archimedes 4
Quadric Surfaces 7
A Personal Address Book 10
Barry Christie Visuals
The Giant Scroller 14
Acorn Launch Archimedes
The Software 16
Mode 7 Screen Editor 23
BEEBUG Education 27
The Master Pages
Turbo ADFS Menu 37
Master Hints 40
New Master Cartridges 41
First Course
Plotting For Effect (Part 2) 44
How to Write an Adventure Game (Part 2) 46
Jungle Adventure
Exploring Assembler (Part 2) 51
Midas Update 54
Workshop
The Accumulation of Errors 58

REVIEWS

Intelligent Modems 20

REGULAR ITEMS

Editorial Jottings 3
News 6
Points Arising 15
Supplement 29-36
Postbag 60
Hints & Tips 61

HINTS AND TIPS

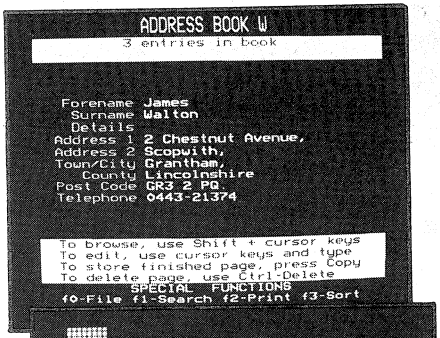
GENERAL

Inter-Word & Long Commands
Word Processing Basic Programs
Dates in Wordwise
Self Destruct Function Keys
Crosswords with Spellmaster
Two Programs in Memory at Once

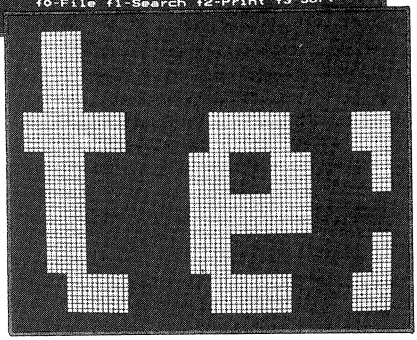
MASTER

Automatic date Stamping (ADFS)
EXEC Files Update Updated
Clearing RAM Banks
Editorial Concatenating

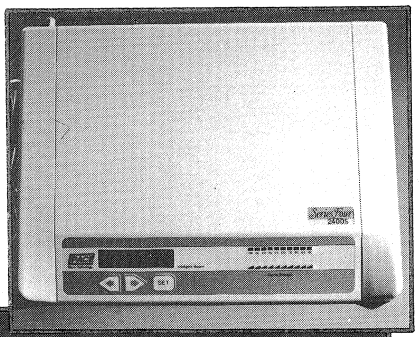
Address Book



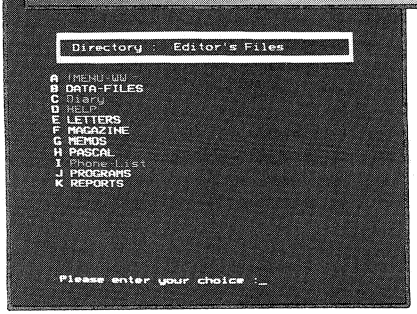
Giant Scroller



Intelligent Modems



Turbo ADFS Menu



EDITORIAL JOTTINGS

HAIL ARCHIMEDES

Whereas the Master showed the 8 bit technology of the BBC micro being stretched to its logical maximum, the launch of Acorn's new Archimedes range represents a total re-think and re-design. This design philosophy leapfrogs all existing 16 bit micros, through its combination of 32 bit architecture and RISC technology, to provide what Acorn claim is the fastest micro in the world. And yet Archimedes manages to retain strong family links with all existing BBC micros through its continued use of BBC Basic in particular, and the retention of a Beeb-like operating system and ADFS. Thus Acorn would appear to have achieved the best of both worlds. On its technical achievement, Acorn are to applauded.

There are, however, two questions which one must ask about Archimedes. Firstly, will the retention of the BBC style operating system prove to be a crucial flaw in an otherwise brilliant design? Can Acorn afford to ignore the rest of the world (MS-DOS for example) as they have successfully done in the past? Acorn clearly believe so, yet many will wish Acorn had perhaps swallowed its pride and offered full MS-DOS emulation as well.

Secondly, has Acorn pitched the price of the new range at the right level. Of course there will be some who will always say a new micro is too expensive, but many home users may well be dismayed when they add VAT on to the quoted prices. However, if Acorn's claim to have produced the world's fastest micro is true then there are many inferior machines substantially more expensive than Archimedes, and these more expensive machines will continue to sell well because they also provide excellent software solutions to real problems. In the end, the success of Archimedes may well depend on the development of wide ranging high-quality software, and the price of the hardware will then be less important.

BEEBUG SUPPORT FOR ARCHIMEDES

We believe that Archimedes is an outstanding computer with enormous potential, and as the newest BBC micro it will have BEEBUG's fullest support. However, we are well aware of the very large number of contented model B users, and those likewise with a Master or Compact. In order that we may give adequate attention to Archimedes without diminishing our support for existing users we will be increasing the size of the magazine from the next issue. Whatever your machine, we intend that BEEBUG shall continue to provide the best possible support through both the magazine and the other benefits of BEEBUG membership.

PROGRAM/REVIEW CLASSIFICATION

Computer System		Filing System	
Master (Basic IV)	M	ADFS	A
Compact (Basic IV)	C	DFS	D
Compact (Basic VI)	C	Cassette	
Model B (Basic II)	B	**Tube Compatibility**	
Model B (Basic I)	B	Tube	
Electron	E		

ACORN LAUNCH ARCHIMEDES

As forecast last month, Acorn has launched the Archimedes, a new range of RISC-based super-micros. Lee Calcraft and Mike Williams give you a first appraisal of these very fast and exciting machines.

The 16th June may have marked the dawn of a new era in micro-computing, with Acorn's launch of a new generation of computer systems based on RISC technology. Named 'Archimedes', this range comprises two separate series of machines; the A305 and A310, and the A410 and A440. Initially, only the BBC branded 300 series is available, with low volume production through the summer building up to full volume from September onwards. The 400 series will be available from late autumn through into 1988, and will carry only the Acorn brand name, not the BBC's. The arrival of Archimedes also places a question mark against the future of the Master and Compact. However, Acorn has also announced price reductions of £50 on the Master and £100 on the Compact, and says there are no plans to discontinue production of either model.

When the model B was first launched in late 1981, it soon established itself as a market leader in education, in the home market and elsewhere. Acorn are now more selective, identifying specific 'niche' markets for targetting new products. In the case of Archimedes, the education market is top of Acorn's priority list. It is here that Acorn will judge their success, regardless of what others say.

This overview (it would be impossible to describe all the features of the new systems in detail in one issue) of the Archimedes range is concerned primarily with the A300 series, the 'new' BBC micro. In this part, Mike Williams describes the new range, while elsewhere in this issue, Lee Calcraft takes a more detailed look at the software supplied with Archimedes.

Archimedes with Desktop Display

Archimedes Specifications

A305 0.5 Mbytes RAM
1.0 Mbytes integral 3.5" floppy disc
512K ROM
Red 'BBC' function keys (12 in all)
Analogue colour monitor connection
Monochrome composite video socket
Parallel printer port
Serial RS423 port
Stereo minature jack socket
Econet socket (interface extra)

A310 As for A305 but 1 Mbyte RAM

A410 1.0 Mbytes RAM
1.0 Mbytes integral 3.5" floppy disc
512K ROM
IBM style keyboard
Other connections as 305
Four socket backplane
Winchester disc interface

A440 As for A410 but 4 Mbytes RAM
Integral 20 Mbyte Winchester

The Archimedes range is based on the latest (2 micron) version of the ARM chip set (see the article in last month's BEEBUG - Vol.6 No.2 - for a discussion of RISC technology and the ARM chip set). The

result is a truly staggering speed increase over existing BBC micros to produce what Acorn claim is the fastest microcomputer in the world.

The new machines are totally different in appearance to existing BBC micros. Gone is the BBC 'beige'! Instead, all the systems follow a standard 'three box' approach, superficially similar to the Compact, but in off-white with the Archimedes logo in blue. The keyboard is a completely new low profile design which generally follows the IBM PC standard, though the A300 'BBC' series retain the red function keys of the BBC micro. The keyboard is a thoroughly professional product, with many extra key functions in addition to the standard QWERTY layout, numeric keypad and function keys. The one blemish in my opinion is the perspex keystrip holder, in the form of a hinged slot, which looks clumsy in comparison with the rest of the keyboard.

Archimedes A300 Series Keyboard

All Archimedes micros are supplied with a three-button mouse that plugs into the keyboard. This has a good feel to it, and is well designed for WIMPs operations. The keyboard in turn plugs into the main system box which also acts as a monitor stand. This box contains the main circuit board with the four ARM chips, memory and other components, substantial power supply, and integral 3.5" floppy drive. This is mounted at a slight angle for easy insertion and removal of discs.

A second floppy drive can be mounted alongside the first; alternatively this slot can house a 20 Mbyte Winchester drive, though this needs an additional interface. This takes the form of a 'podule', Acorn's new term for a peripheral module. Podules, in turn must be mounted in a back plane, and a two-socket back plane (and a Winchester podule) will be available from September as enhancements to the A300 series. Other interfaces to be provided include an I/O podule, a ROM podule, and maybe a MIDI podule.

Archimedes Prices

A305	£799	(£918.85) without monitor
	£849	(£976.35) with mono. monitor
	£999	(£1148.35) with colour monitor
A310	£875	(£1006.25) without monitor
	£925	(£1063.25) with mono. monitor
	£1075	(£1236.25) with colour monitor

Memory upgrade A305 to A310 £89 (£102.35)

Prices in brackets include VAT

All connectors are positioned at the rear of the processor box, and are thus easily accessible - no more groping under the machine but then there is no user port, bus or analogue connection. The box itself is of solid metal construction except for the protruding plastic moulding at the front enclosing the disc drive. I do not yet feel quite happy with this design, and would have preferred a simple flat front to the system, but that's just a personal view.

As for monitors, the same monochrome and colour monitors will be supplied with Archimedes as with the Compact, but using an analogue (minature 9-pin D-shape) connector. The monochrome monitor offers high resolution, the colour monitor medium resolution. This latter does not really produce a clear enough image for anyone working at length in any of the 80 column modes (let alone the 132 column modes), which is a pity, as many enthusiasts fired by the colour and graphics capabilities of the new machines will undoubtedly be tempted by the colour option.

Apart from a huge increase in performance (a short Basic program to multiply a cosine by a sine 10000 times took 517 seconds to run on a standard model B, just 9.1 seconds with RISC!), the Archimedes machines provide, obviously, far more RAM, but also up to 256 colours on screen selectable from a palette of 4096, and an eight-channel seven-position stereo sound system.

→16

Wordwise for Ever

Norwich Computer services have announced Word-Ex, the Wordwise Users' Extension ROM. This works with both Wordwise and Wordwise Plus to provide most of the features of Inter-Word, but goes way beyond Inter-Word by providing high speed sorting, label-printing, case changing, alphabetic file access and function key programming, and all for £30. Word-Ex is available from NCS at 18 Mile End Road, Norwich NR4 7QY, or phone Norwich (0603) 507057.

Wordwise Plus II is the new release from Interface Electronics (IFEL) who with CC's co-operation have produced a 32K version of the popular wordprocessor with more than a dozen additional Ctrl functions. Wordwise Plus II is a complete replacement for WW+ and costs £65.95 direct from IFEL. Discounts are available for existing Wordwise Plus users. IFEL are on Plymouth (07555) 7286.

Repton Goes Global

Prolific games software house Superior has released 'Around the World in 40 Screens', a collection of 40 highly entertaining new screens for Repton 3. Both the Repton 3 program and editor are supplied as part of the new package which costs £6.95 on cassette (BBC and Electron), £7.95 for 5.25" disc and £9.95 for 3.5" disc (for the Compact). Superior, as always, are on (0532) 459453.

Inter-Base Arrives

The final ROM in the Inter-Link series from Computer Concepts, the database package Inter-Base, is now available at long last. This is more of a language than a database and may be used to control the operation of the other ROMs in the Inter-Link series. Inter-Base costs £67.85 and Computer Concepts are on (0442) 63933. Inter-Base is also available from BEEBUG Mail Order (see catalogue).

BT Rewards Hackers

Amongst latest offerings from BT subsidiary Firebird Silver is The Hacker for the BBC micro. Crack the telephone network and hack your way into the computer games library. What BT think of this is not known. 'Birdstrike', another BBC game from Firebird, is a typical 'shoot-em-up' game where you pilot an aircraft attacking enemy planes and dropping bombs. Both games are at the budget price of £1.99 from retailers, or phone 01-379 6755.

BEEBUG First

The superb BEEBUG Master printer buffer utility (see BEEBUG Vol.5 No.7) has now been extended by author Tim Powys-Lybbe to work for the B+128, Compact and any Beeb with sideways RAM. This is being marketed (at a price) by Care Electronics, but the original version is, of course, free to BEEBUG members for no more than the the price of a back copy. A ROM version is in regular use in the magazine office and has proved invaluable and very reliable.

It's all Greek to Minerva

After the success of its database generator package called System Delta (see review in BEEBUG Vol.5 No. 4), software house Minerva Systems have announced another world first for the Beeb. System Gamma is a programmable graphics package which allows the non-programmer to generate business charts and graphics of a sophistication unrivalled on the BBC micro. As many charts as you wish may be displayed on the screen at any one time with a choice of scatter, histogram, line and pie chart formats.

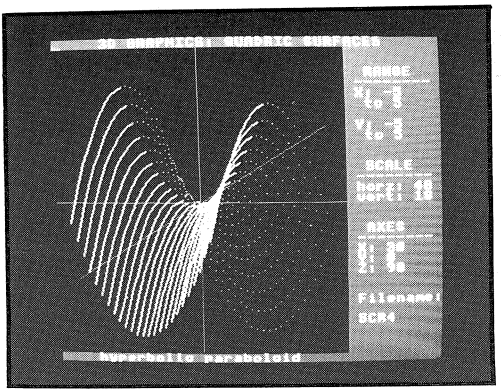
System Gamma is written in the same style as the successful System Delta, and will accept data from System Delta and other sources. System Gamma costs £45.95 and Minerva are on Exeter (0392) 37756.

Rural Rides

Soft-Teach has released games versions of its highly praised educational map-reading packages. The new versions, Rural Rambles and Seaside Strolls, provide endless challenges to your map-reading skills. Both packs are on disc for the BBC B, B+ and Master, and cost £10.95 each inclusive direct from Soft-Teach, Sturgess Farmhouse, Longbridge Deverill, Warminster, Wilts or phone (0985) 40329.

QUADRIC SURFACES

Exciting three-dimensional images are the result of Jan Stourman's latest program. Apply this technique to your own program and smarten up those graphic displays.



Three-dimensional graphics on a computer screen are always fascinating to watch. Wire-frame graphics are the most popular, and computationally the easiest way of picturing a graph in three dimensions. However, ordinary wire-frame pictures are often confusing to the viewer, as it is often difficult to see which lines are in the foreground and which are behind. Removing hidden lines is one way of solving this ambiguity, and a nice program to do this was published in BEEBUG Vol.3 No.8 by Q.A.Rice. The price we usually pay for removing hidden lines is the loss of all details of the hidden parts of the surface.

The method used here, in the program Quadric, also creates an illusion of depth, but retains some details of the parts in the background. This is achieved by a combination of colour and dot density. The part of the surface that is "closest" to the viewer is more brightly coloured and its dots are connected by lines (using PLOT5). Points in the background are separated (using PLOT69) and coloured less brightly.

RUNNING THE QUADRICS PROGRAM

To illustrate the power of this method, data for seven different surfaces is included as part of the Quadric program listed below. Type in the program and save to disc or tape before running. After choosing a surface from the menu, the user is asked to enter values for the co-ordinate angles. The default values are 30, 0 and 90 degrees for the x, y and z-axes, respectively. Pressing Return leaves the angles of the co-ordinate axes unchanged. The user is finally asked whether or not the axes are to be drawn.

The values of the most important parameters are shown in a text area on the right of the screen, leaving an almost square graphics window on the left. Once plotting of a surface is completed, the user is offered a number of options. Screens may be dumped to a printer, using any suitable mode 1 screen dump routine. Alternatively, the screen may be saved to disc for quick *LOADing. Remember to use the same colour assignments (VDU19) before reloading previously saved screens. Pressing the Escape key at any time will take you back to the menu screen. Option 'Q' will quit the program.

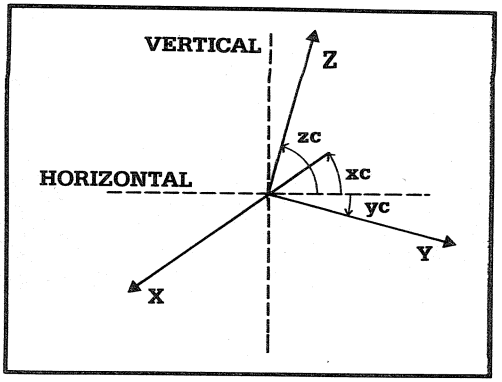
You can readily experiment with this program by varying any of the parameters contained in the data statements (lines 10000 to 10630), or the functions themselves in the same lines. If you want to try alternative colour combinations, then change the VDU19 statements at lines 1020 and 1030, and the assignments at line 1040. Users with monochrome screens may well find this worthwhile.

MATHEMATICAL NOTES

The first six sets of data statements generate the main types of quadric surface (derived from quadratic equations). All other (non-degenerate) quadric surfaces are derived from these by rotation, translation and scaling. The seventh set of data statements creates a non-quadric surface around a multiple saddle point.

The position of the co-ordinate axes is determined by their angle relative to the BBC micro's horizontal and vertical axes as in the following diagram. Positive

angles are rotated anti-clockwise. The orientation of the x-axis is later reversed by the program so that positive x-values lie "towards" the viewer.



PROGRAM NOTES

The last set of DATA statements (lines 10700-10720) shows the general format. The user can easily add new graph specifications using this format. Note, however, that the last data statement must contain a "?" as at line 10700, and that sets of data start at consecutive hundreds, i.e. 10200, 10300, etc. When experimenting with new graphs, choose large values for xstp and ystp initially. Once the co-ordinate end points and scaling factors are determined, the step values can be adjusted to give the clearest picture.

If the expression of the graph includes the function SQR, the variable S% is set to TRUE. Lines 2200-2240 then extract the argument of the SQR function, arg$, from the expression by balancing brackets: when the number of opening and closing brackets after SQR are equal, arg$ is contained between the outermost set of brackets and MID$ is applied.

Similar treatment will be necessary for expressions that contain other Basic functions with restricted domains. For the functions LOG and LN line 6060 becomes:

```
6060 IF S% IF EVALarg$<=0 GOTO 6100
```

while for ACS or ASN we get:

```
6060 IF S% IF EVALarg$<-1 OR
     EVALarg$>1 GOTO 6100
```

It is, of course, also necessary to change the SQR in line 2190 to the new Basic function keyword. In all these cases lines 6020 and 6120 must be omitted from the program, and line 6070 simplifies to:

```
6070 Z=EVAL(expr$)
```

When the expression includes SQR, both branches of the surface are plotted corresponding to the positive (CHR$43) and the negative (CHR$45) value of the square-root function. This is achieved most easily by the *REPEAT-UNTIL* construct as in lines 6020 and 6120.

```
   10 REM program QUADRIC
   20 REM Version B1.4
   30 REM Author  Jan Stuurman
   40 REM BEEBUG  July 1987
   50 REM program subject to copyright
   60 :
  100 MODE1:ON ERROR GOTO 900
  110 PROCinit:DIM os 40
  120 PROCmenu
  130 PROCaxes:PROCparam
  140 PROCsurface:PROCoption
  150 MODE7:*FX4,0
  160 END
  170 :
  900 IF ERR=17 GOTO120
  910 VDU26,23,1,255;0;0;0;:*FX4,0
  920 REPORT:PRINT" at line ";ERL
  930 END
  940 :
 1000 DEFPROCinit:*FX4,1
 1010 VDU23,1,0;0;0;0;
 1020 VDU19,0,0;0;19,1,4;0;
 1030 VDU19,2,5;0;19,3,7;0;
 1040 black=0:blue=1:magenta=2:white=3
 1050 COLOUR128+white:CLS:COLOUR blue
 1060 PRINTTAB(5)"3D GRAPHICS: QUADRIC S
URFACES"
 1070 VDU24,0;32;956;988;29,480;512;
 1080 L=476:GCOL0,128+black:Q%=0
 1090 xc=30:yc=0:zc=90:dr=PI/180
 1100 ENDPROC
 1110 :
 2000 DEFPROCmenu LOCALB%,G%,N%,O%,P%
 2010 VDU28,0,31,39,1,12:COLOUR blue
 2020 PRINTTAB(15,2)"GRAPH MENU":N%=-1
 2030 REPEAT N%=N%+1:RESTORE (10000+100*
N%)
 2040 READname$:IF name$="?" GOTO 2060
 2050 PRINTTAB(8,4+N%)name$
 2060 UNTIL name$="?":COLOUR black
 2070 PRINTTAB(2,5+N%)"Use CURSOR (UP/DO
WN) keys and RETURN"
 2080 PRINTTAB(11)"to select option."
 2090 REPEAT
 2100 PRINTTAB(5,4+O%)SPC2;TAB(5,4+Q%)">
>";
 2110 O%=Q%:G%=GET
 2120 IF G%=138 Q%=(O%+1)MODN%
 2130 IF G%=139 Q%=(O%+N%-1)MODN%
 2140 UNTIL G%=13
```

```
2150 RESTORE (10000+100*O%)
2160 READ name$,expr$
2170 READ xmin,xmax,xstp,ymin,ymax,ystp
2180 READ hscale,vscale
2190 P%=INSTR(expr$,"SQR"):IF P%=0 S%=F
ALSE:ENDPROC
2200 S%=TRUE:arq$=MID$(expr$,P%+3)
2210 B%=0:P%=0:REPEAT P%=P%+1
2220 IF MID$(arg$,P%,1)="(" B%=B%+1
2230 IF MID$(arg$,P%,1)=")" B%=B%-1
2240 UNTIL B%=0:arg$=MID$(arg$,2,P%-2)
2250 ENDPROC
2260 :
3000 DEFPROCaxes LOCALG%
3010 VDU28,0,30,39,20,12:COLOUR blue
3020 PRINTTAB(4,0)"Enter angles of coor
dinate axes"
3030 PRINTTAB(4)"or <RETURN> to leave u
nchanged."
3040 COLOUR black:PRINTTAB(4,3)"ALL ANG
LES ARE GIVEN IN DEGREES":COLOUR blue
3050 PRINTTAB(14,5)"X-axis:";:xc=FNi(xc
)
3060 PRINTTAB(14,6)"Y-axis:";:yc=FNi(yc
)
3070 PRINTTAB(14,7)"Z-axis:";:zc=FNi(zc
)
3080 XC=xc*dr+PI:YC=yc*dr:ZC=zc*dr
3090 sx=SIN(XC):sy=SIN(YC):sz=SIN(ZC)
3100 cx=COS(XC):cy=COS(YC):cz=COS(ZC)
3110 PRINTTAB(6,9)"Draw coordinate axes
? (Y/N)"
3120 REPEAT G%=GETAND&5F:UNTIL G%=78 OR
G%=89
3130 VDU28,0,31,39,1,12:CLG
3140 IF G%=78 ENDPROC ELSE GCOL0,white
3150 MOVE L*cx,L*sx:DRAW -L*cx,-L*sx
3160 MOVE L*cy,L*sy:DRAW -L*cy,-L*sy
3170 MOVE L*cz,L*sz:DRAW -L*cz,-L*sz
3180 ENDPROC
3190 :
4000 DEFFNi(A) LOCALG%,X%,Y%,G$
4010 X%=POS+1:Y%=VPOS:PRINT" ";A
4020 VDU31,X%,Y%:*FX15,1
4030 G%=GET:IF G%=13 =A
4040 PRINTSPC8:VDU31,X%,Y%,G%:G$=CHR$G%
4050 REPEAT G%=GET
4060 IF G%=127ANDLENG$>0 G$=LEFT$(G$,LE
NG$-1):VDUG%
4070 IF G%>45ANDG%<58 G$=G$+CHR$G%:VDUG
%
4080 UNTIL G%=13:=VALG$
4090 :
5000 DEFPROCparam
5010 PRINTTAB(15-LENname$/2,30)name$;
5020 VDU28,31,24,39,3:d$=STRING$(8,"-")
5030 PRINT" RANGE"'d$
5040 PRINTTAB(0,2)"X: ";xmin'" to ";xma
x
5050 PRINTTAB(0,5)"Y: ";ymin'" to ";yma
x
5060 PRINTTAB(0,9)" SCALE"'d$
5070 PRINTTAB(0,11)"horz: ";hscale;
5080 PRINTTAB(0,12)"vert: ";vscale;
5090 PRINTTAB(0,15)" AXES"'d$
5100 PRINTTAB(0,17)"X: ";xc'"Y: ";yc'"Z
: "zc
5110 ENDPROC
5120 :
6000 DEFPROCsurface LOCALX,Y,Z,D%
6010 fore=blue:back=magenta
6020 sign=41:REPEAT sign=sign+2
6030 FORY=ymin TO ymax STEPystp
6040 FORX=xmin TO xmax STEPxstp
6050 IF X<0 D%=69:GCOL0,back
6060 IF S% IF EVALarg$<0 GOTO 6100
6070 Z=EVAL(CHR$sign+expr$)
6080 PLOTD%,(X*cx+Y*cy+Z*cz)*hscale,(X*
sx+Y*sy+Z*sz)*vscale
6090 IF X>=0 D%=5:GCOL0,fore
6100 NEXT
6110 NEXT
6120 UNTIL NOT S% OR sign=45
6130 ENDPROC
6140 :
7000 DEFPROCoption LOCALG%
7010 REPEAT VDU28,31,31,39,25,12
7020 PRINT" OPTION"'d$
7030 PRINT"COPY:dumpTAB :saveESC :menuQ
  :quit";
7040 REPEAT G%=GET:UNTIL G%=135 OR G%=9
OR G%=81
7050 IF G%=135 PROCdump
7060 IF G%=9 PROCsave
7070 UNTIL G%=81:ENDPROC
7080 :
8000 DEFPROCdump CLS
8010 REM INSERT CALL TO YOUR
8020 REM SCREEN DUMP ROUTINE
8030 ENDPROC
8040 :
9000 DEFPROCsave CLS
9010 INPUT"Filename:"'fn$
9020 PROCoscli("SA."+fn$+" 3000+4FFF")
9030 ENDPROC
9040 :
9500 DEF PROCoscli($os):LOCAL X%,Y%
9510 X%=os:Y%=os DIV 256:CALL &FFF7
9520 ENDPROC
9530 :
10000 DATA paraboloid,X*X+Y*Y-40
10010 DATA -6,6,.4,-6,6,.4
10020 DATA 40,10
10030 :
10100 DATA ellipsoid,SQR(36-X*X-Y*Y)
10110 DATA -6,6,.15,-6,6,1
10120 DATA 35,45
10130 :
```

→50

A Personal Address Book

If you make frequent use of your micro, what better than to use it as your personal address book as Simon Williams explains. Now all those addresses and phone numbers will always be to hand.

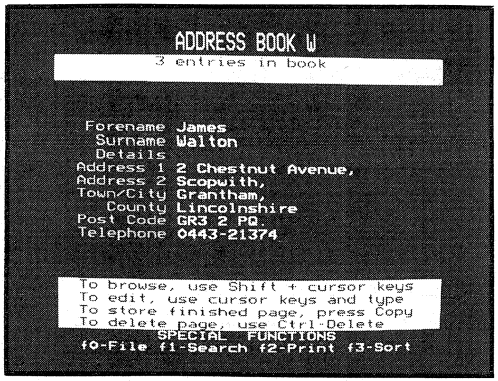
Since the introduction of the BBC Micro there have been dozens of databases designed to store facts and figures on your micro. Nearly all of these have been designed with a host of facilities, which make them very flexible but quite cumbersome to use for simple applications.

For something as straightforward as storing names and addresses, you really want a program which is very quick and easy to use, but can still hold enough information to make it worthwhile. The program presented here can hold over 2000 entries, but is very easy to use, offers searching, sorting and printing facilities, and takes only just over 6K of memory.

It is structured around the idea of 26 'books' of addresses, labelled 'A' to 'Z', plus a special book '?', which is used when searching any of the others. Each book can hold up to 75 entries, and these are held in memory all the time the program is running, so access to any entry is virtually instantaneous. Switching books only takes a couple of seconds, as they are loaded and saved from disc.

The program works from a mode 7 screen which displays a single name and address, prompt lines and instructions at all times. The controls are kept very simple and are mostly based around the cursor keys and the first four function keys.

When you run the program, the address area in the centre of the screen is initially blank, apart from the 'Surname', 'Telephone No' prompts etc. At the bottom of the screen are a set of instructions which stay on-screen all the time. At the top is the program title, and below that are two command lines where prompts and responses appear.



To enter a name and address, just type away. Use Return to move from line to line, and the cursor keys to move around the entry area (which can be thought of as a continuous area) to the right of the name and address prompts. Entering text here has no effect on any entry in the current book, until you press Copy, which copies it into memory. It is ESSENTIAL to press Copy when you've updated an existing entry, before moving to another, or the update won't be saved into a book. The program may be terminated at any time by pressing Shift-Escape, but your latest screen will not have been copied to the address book unless you have pressed Copy.

You may also use Shift with the left and right cursor keys to move to the previous entry or to the next entry, and with the up and down cursor keys to move to the first entry or last entry.

When you've entered a few names, you'll probably want to save the book to disc. Press the function key f0 and follow the prompts to give the book an identification letter. When you next run the program, you can re-load the same book by quoting its letter.

Sorting a book is simply a matter of selecting the field you want to use as a key and waiting a short while for the sort to take place. The program uses the same type of 'exchange sort' technique that is used on the 'Welcome' tape.

Searching follows much the same pattern as sorting, but you must enter the word or phrase you want to find, as well as the

field in which to search. You can either display each matching entry on the screen, or send them all to the special book '?', which will then effectively contain a subset of the book. You can load book '?' in exactly the same way as any other, and further refine your search using another field. Searching is case-specific and will find part entries. For example, a search for 'Smi' will find Smit, Smith, and Smithson.

You can print entries from a book on any Epson compatible printer, in either of two ways. The report option will print a list of all entries straight down the paper, while the label option formats just the names and addresses correctly for single labels. With the label option, you can select individual names to print by browsing through the entries. To alter the number of lines per label, try experimenting with the 9 in line 2480 of PROCprintone.

As far as the listing goes, the main program finishes at line 270, and the rest of the code is a series of procedures tied to specific features of the program's function. Control of the screen is mainly by VDU statements, with extensive use made of windows to delimit entry areas.

The names and addresses in a book are all held in one large array, A$, in memory and each new entry is read directly from the screen to save space. The limit of 75 entries per book relies on the fact that most entries will only occupy a small part of the array space allocated to them. If your entries are likely to be particularly long, it would be wise to alter the value: epb=74 found at line 1010 in PROCsetup, to a lower value.

```
  10 REM Program Address Book
  20 REM Version B1.7
  30 REM Author  Simon Williams
  40 REM BEEBUG  July 1987
  50 REM Program subject to copyright
  60 :
 100 MODE7:ON ERROR GOTO 290
 110 PROCsetup
 120 PROCshowpage
 130 REPEAT
 140 IF INKEY-58 IF INKEY-1 IF R%>0 R%=
0:PROCshowpage
 150 IF INKEY-42 IF INKEY-1 R%=T%-(T%<e
pb):PROCshowpage
 160 IF INKEY-26 IF INKEY-1 IF R%>0 R%=
R%-1:PROCshowpage
 170 IF INKEY-122 IF INKEY-1 IF R%<=T%
R%=R%+1:PROCshowpage
 180 IF INKEY-90 IF INKEY-2 IF T%>0 FOR
 Y%=R% TO T%-1:FOR X%=0 TO 8:A$(X%,Y%)=A
$(X%,Y%+1):NEXT,:T%=T%-1:PROCshowpage
 190 IF INKEY-106 IF T%<epb PROCstore
 200 IF INKEY-106 IF T%>epb-1 PROCwl(0)
:PRINTTAB(11,2)R$;F$"   FILE  FULL";SPC5
PROCw2(0):VDU7
 210 IF INKEY-33 PROCfile
 220 IF INKEY-114 IF FNt THEN PROCsearch
 230 IF INKEY-115 IF FNt THEN PROCprint
 240 IF INKEY-116 IF FNt THEN PROCsort
 250 PROCedit
 260 UNTIL FALSE
 270 END
 280 :
 290 VDU26:*FX21,3
 300 IF ERR=198 CLOSE#F%:PRINTTAB(8,2)"
 Disc full - Compacting":*COMPACT
 310 IF ERR=201 I$=FNchoice("Read only
disc - change it",s$,s$,7)
 320 IF ERR=190 I$=FNchoice("Too many f
iles - change disc",s$,s$,6)
 330 IF ERR<>17 OR NOT INKEY-1 PROCwl(1
):R%=0:L%=1:GOTO 120
 340 ON ERROR OFF
 350 MODE7:*FX4,0
 360 END
 370 :
1000 DEF PROCsetup
1010 epb=74:R%=0:T%=0:L%=1:DIM C$(8),T$
(8),A$(8,epb),os 40:*FX4,1
1020 g$="?":s$=CHR$32:R$=CHR$129:Y$=CHR
$131:B$=CHR$132:W$=CHR$135
1030 F$=CHR$136:S$=CHR$137:D$=CHR$141:N
$=CHR$157
1040 FOR Y%=0 TO 18
1050 PRINTTAB(0,Y%)B$;N$;Y$TAB(12,Y%)W$;
1060 NEXT Y%
1070 PRINTTAB(11,0)D$;Y$;"ADDRESS BOOK"
TAB(11,1)D$;Y$;"ADDRESS BOOK"'W$;N$;B$'W
$;N$;B$
1080 FOR Y%=0 TO 8
1090 READ C$(Y%):PRINT TAB(12-LEN(C$(Y%
)),Y%+7)C$(Y%)
1100 NEXT Y%
1110 DATA Forename,Surname,Details,Addr
ess 1,Address 2,Town/City,County,Post Co
de,Telephone
1120 PRINTTAB(0,19)W$;N$;B$;"To browse,
 use Shift + cursor keys."'W$;N$;B$;"To
edit, use cursor keys and type."'W$;N$;B
$"To store finished page, press Copy."
1130 PRINTW$;N$;B$"To delete page, use
Ctrl-Delete."'R$;N$;W$;SPC8"SPECIAL  FUN
CTIONS"'R$;N$;W$;"f0-File f1-Search f2-P
rint f3-Sort";
```

```
1140 ENDPROC
1150 :
1160 DEF PROCedit
1170 I%=INKEY(0)
1180 IF I%=136 IF VPOS>0 OR POS>0 VDU8
1190 IF I%=137 IF VPOS<8 OR POS<25 VDU9
1200 IF I%=138 IF VPOS<8 VDU10
1210 IF I%=139 IF VPOS>0 VDU11
1220 IF I%=13 IF VPOS<8 VDU13,10
1230 IF I%=127 IFVPOS>0 OR POS>0 VDU127
1240 IF VPOS=8 AND POS=26 ENDPROC
1250 IF I%>31 IF I%<127 VDUI%
1260 ENDPROC
1270 :
1280 DEF PROCfile
1290 VDU26,23;11,0;0;0;0
1300 I$=FNchoice("Load or Save book","L
","S",8)
1310 IF I$="L" PROCload ELSE IF T%>0 TH
EN PROCsave ELSE B%=0
1320 PRINTTAB(26,0);:VDUB%,10,8,B%
1330 PROCwl(1):PROCshowpage
1340 ENDPROC
1350 :
1360 DEF PROCload
1370 IF T% I$=FNchoice("Save current bo
ok first","Y","N",6):IF I$="Y" PROCsave
1380 T$=FNgetbook("  Load")
1390 PRINTTAB(3,2)F$SPC9"Loading";S$;"b
ook";R$;T$;SPC8TAB(17,7);
1400 F%=OPENUP(T$+"-BOOK")
1410 IF F%=0 I$=FNchoice("No such book"
,s$,s$,13):B%=0:ENDPROC
1420 INPUT#F%,T%
1430 FOR Y%=0 TO T%-1
1440 FOR X%=0 TO 8
1450 INPUT#F%,A$:A$(X%,Y%)=FNstrip(A$)
1460 NEXT,
1470 CLOSE#F%:R%=0
1480 ENDPROC
1490 :
1500 DEF PROCsave
1510 T$=FNgetbook(" Save as")
1520 PRINTTAB(3,2)F$;SPC10"Saving";S$;"
book";R$;T$;SPC10TAB(17,7);
1530 F%=OPENIN(T$+"-BOOK")
1540 IF F% THEN CLOSE#F%:OSCLI("DELETE
"+T$+"-BOOK")
1550 F%=OPENOUT(T$+"-BOOK")
1560 PRINT#F%,T%
1570 FOR Y%=0 TO T%-1
1580 FOR X%=0 TO 8
1590 PRINT#F%,A$(X%,Y%)
1600 NEXT,
1610 CLOSE#F%
1620 ENDPROC
1630 :
1640 DEF FNgetbook(T$)
1650 PRINTTAB(3,2)T$;" which book";R$;"
(A to Z or "g$")     "
1660 REPEAT:B1%=GET:B%=B1% AND &5F:UNTI
L B1%=ASCg$ OR (B%>64 AND B%<91)
1670 IF B1%=ASCg$ THEN B%=ASCg$
1680 =CHR$B%
1690 :
1700 DEF PROCsearch
1710 VDU12,23;11,0;0;0;0
1720 Q$=FNchoice("Display or Store in b
ook '"+g$+"'","D","S",3)
1730 Y%=FNgetfield("  Search on")
1740 PRINTTAB(3,2)"Search on";R$;C$(Y%)
;SPC17;TAB(9,3)"for";R$;
1750 VDU23;11,255;0;0;0:T$="":*FX15,1
1760 REPEAT
1770 I%=GET
1780 IF I%=127 IF POS>13 VDU127:T$=LEFT
$(T$,LEN(T$)-1)
1790 IF POS<38 IF I%>31 IF I%<127 VDUI%
:T$=T$+CHR$(I%)
1800 UNTIL I%=13
1810 VDU23;11,0;0;0;0
1820 IF Q$="S" PROCsavesearch:ENDPROC
1830 M%=0:FOR R%=0 TO T%-1
1840 IF M% IF INSTR(A$(Y%,R%),T$) VDU26
:PRINTTAB(3,2)F$;"MORE";S$;"matches
";R$;"(Press any key)"TAB(13,7);:IFGET:P
ROCshowpage:M%=R%+1 ELSE IF INSTR(A$(Y%,
R%),T$) PROCshowpage:M%=R%+1
1850 NEXT R%
1860 IF M%=0 R%=0:PRINTTAB(3,2)"No matc
h found    ";R$;"(Press any key)":IFGET:P
ROCshowpage ELSE R%=M%-1
1870 PROCwl(1):PROCw2(0)
1880 ENDPROC
1890 :
1900 DEF PROCsavesearch
1910 M%=0:FOR R%=0 TO T%-1
1920 IF INSTR(A$(Y%,R%),T$) M%=M%+1
1930 NEXT R%
1940 IF M%=0 R%=0:PRINTTAB(3,2)"No matc
h found    ";R$;"(Press any key)":IFGET:P
ROCwl(1):PROCshowpage:ENDPROC
1950 PRINTTAB(3,2)R$;M%;B$"matches foun
d  ";F$;"Saving";S$;"in book";R$;g$
1960 F%=OPENIN(g$+"-BOOK"):IF F% THEN C
LOSE#F%:PROCoscli("DELETE "+g$+"-BOOK")
1970 F%=OPENOUT(g$+"-BOOK")
1980 PRINT#F%,M%
1990 FOR R%=0 TO T%-1
2000 IF INSTR(A$(Y%,R%),T$) FOR X%=0 TO
8:PRINT#F%,A$(X%,R%):NEXT
2010 NEXT
2020 CLOSE#F%:R%=0
2030 PROCwl(1):PROCshowpage
2040 ENDPROC
2050 :
2060 DEF FNgetfield(T$)
2070 PRINTTAB(3,2)T$;" which field";R$;
"(0 to 8)  "
2080 VDU28,13,16,13,7
```

```
2090 VDU23;11,0;0;0;0
2100 PRINT"Ø12345678";
2110 REPEAT:Y%=GET-48
2120 UNTIL Y%>-1 AND Y%<9
2130 VDU12,26
2140 =Y%
2150 :
2160 DEF PROCprint
2170 VDU26,23;11,0;0;0;0
2180 I$=FNchoice("Report or Labels","R"
,"L",9)
2190 IF I$="L" PROClabels:PROCwl(1):PRO
Cshowpage:ENDPROC
2200 I$=FNchoice("Check printer",s$,s$,
13):*FX3,1Ø
2210 PRINT"ADDRESS BOOK ";CHR$(B%)''
2220 FOR Y%=Ø TO T%-1
2230 PRINTA$(Ø,Y%);s$;A$(1,Y%)
2240 IF A$(2,Y%)>"" PRINTA$(2,Y%)
2250 FOR X%=3 TO 7
2260 IF A$(X%,Y%)>"" PRINT;A$(X%,Y%);",
";
2270 NEXT X%:PRINT;A$(8,Y%)'
2280 NEXT Y%:*FX3,Ø
2290 R%=Ø:PROCwl(1):PROCshowpage
2300 ENDPROC
2310 :
2320 DEF PROClabels
2330 I$=FNchoice("Whole book or Selecte
d pages","W","S",3)
2340 IF I$="W" I$=FNchoice("Check print
er",s$,s$,13):FOR R%=Ø TO T%-1:PROCprint
one:NEXT R%:ENDPROC
2350 PRINTTAB(3,2)R$;"Use [] to Select
COPY=Print RET=End":R%=Ø:L%=Ø:PROCshowpa
ge
2360 REPEAT
2370 I%=GET
2380 IF I%=135 PROCprintone
2390 IF I%=136 IF R%>Ø R%=R%-1:PROCshow
page
2400 IF I%=137 IF R%<T% R%=R%+1:PROCsho
wpage
2410 UNTIL I%=13:L%=1
2420 ENDPROC
2430 :
2440 DEF PROCprintone:*FX3,1Ø
2450 PRINTA$(Ø,R%);s$;A$(1,R%)
2460 I%=1:FOR X%=2 TO 7
2470 IF A$(X%,R%)>"" PRINTA$(X%,R%):I%=
I%+1
2480 NEXT:FOR X%=1 TO 9-I%:PRINT:NEXT
2490 *FX3,Ø
2500 ENDPROC
2510 :
2520 DEF PROCsort
2530 VDU12,26
2540 Y%=FNgetfield("   Sort by")
2550 PRINTTAB(5,2)SPC5;F$;"Sorting";S$;
"by";R$;C$(Y%);SPC9
2560 FOR CP%=Ø TO T%-2
2570 FOR SP%=CP%+1 TO T%-1
2580 IF A$(Y%,SP%)<A$(Y%,CP%) FOR N%=Ø
TO 8:T$(N%)=A$(N%,SP%):A$(N%,SP%)=A$(N%,
CP%):A$(N%,CP%)=T$(N%):NEXT
2590 NEXT,
2600 R%=Ø:PROCwl(1):PROCshowpage
2610 ENDPROC
2620 :
2630 DEF PROCshowpage
2640 TIME=Ø:REPEAT UNTIL TIME>1Ø
2650 IF L% VDU26:PRINTTAB(1Ø,2)R$;T%;B$
"entries in book"
2660 PROCw2(1)
2670 FOR X%=Ø TO 8
2680 PRINTTAB(Ø,X%)A$(X%,R%);
2690 NEXT:VDU3Ø:*FX15,1
2700 VDU23;11,255;0;0;0
2710 ENDPROC
2720 :
2730 DEF FNchoice(T$,F$,S$,X%)
2740 PROCwl(1):VDU26
2750 PRINTTAB(X%,2)T$;
2760 IF F$>s$ PRINT;R$;"(";F$;"/";S$;")
" ELSE PRINTTAB(12,3)R$;"(Press space)"
2770 F%=ASC(F$):S%=ASC(S$) AND &5F
2780 REPEAT:I%=GET AND &5F:UNTIL I%=F%
OR I%=S%
2790 =CHR$(I%)
2800 :
2810 DEF PROCwl(f%)
2820 VDU28,3,3,39,2,12*f%
2830 ENDPROC
2840 :
2850 DEF PROCw2(f%)
2860 VDU28,13,15,39,7,12*f%
2870 ENDPROC
2880 :
2890 DEFPROCoscli($os)
2900 LOCAL X%,Y%
2910 IF $os="" ENDPROC
2920 X%=os:Y%=os DIV 256:CALL &FFF7
2930 ENDPROC
2940 :
2950 DEFFNreadch(col,row)
2960 LOCAL A%,c
2970 VDU 31,col,row
2980 A%=135
2990 c=USR(&FFF4)
3000 c=c AND &FFFF
3010 c=c DIV &100
3020 =CHR$c
3030 :
3040 DEF PROCstore
3050 VDU30
3060 FOR row=Ø TO 8:A$(row,R%)="":NEXT
3070 FOR row=Ø TO 8
3080 FOR col=Ø TO 25
3090 c1$=FNreadch(col,row)
3100 A$(row,R%)=A$(row,R%)+c1$
```

→26

BARRY CHRISTIE

Visuals

The Giant Scroller

This month sees the start of a new and exciting series devoted to visual effects of all kinds. We launch it with a Giant Text Scroller.

This scroller has many applications where an eye-catching display is required; and it will scroll in any one of four different type sizes. Text may be any colour, and user-defined characters are easily incorporated. The listing is relatively short, and in fact almost half its length is taken up with specific text messages. These appear in the the early part of the program in the form of calls to PROCmessage, and again as PROCdefine, which is used to define user characters.

When typing in the program, you may if you wish leave out lines 140 to 250, and lines 1120 to 1210. In this state the program will display just a single message. Whether you have typed in the whole program, or just the essential parts, the key to its use is in PROCmessage. You can include as many calls to this procedure as you wish, and as you can see, it has four parameters: the text string, and three numbers which define the colour, height and width respectively.

The colour code is simply a Teletext graphics colour code, since the display works in mode 7. These range from red at 145 to white at 151. The height and width parameters (H and W) each take the value zero or one, the former giving minimum height or width, and the latter maximum. Thus for example:

```
PROCmessage ("MESSAGE",150,1,0)
```

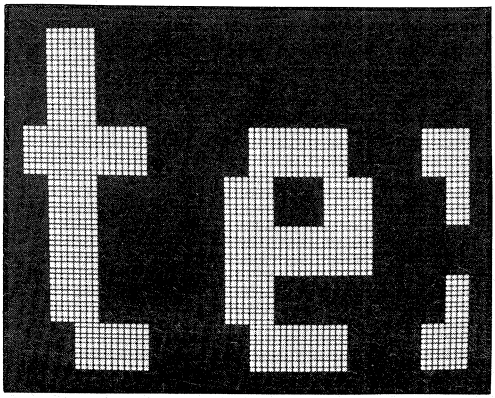
would scroll the word "MESSAGE" in cyan in double height, single width characters.

To see how to incorporate user-defined characters, take a look at lines 200 and 210. The first calls PROCdefine (at line 1120) which defines a number of characters using VDU23, then strings them together in A$. Line 210 then calls PROCmessage with A$ as the first parameter.

The display currently uses mode 7 separated graphics characters. If you wish to use continuous characters, then replace the &9A in line 1300 with &99.

More from me in the next issue. In the mean time, happy scrolling; and remember, if you press Escape to exit the program, you may need to be patient.

```
  10 REM Program Mode 7 Scroll
  20 REM Version B0.7B
  30 REM Author  Barry Christie
  40 REM Beebug  July 1987
  50 REM Program subject to copyright
  60 :
 100 MODE 7:ON ERROR GOTO 1810
 110 PROCinitialise
 120 REPEAT
 130 PROCmessage("This is a demonstrati
on of giant scrolling text on the BBC Mi
cro ..... ",150,1,1)
 140 PROCmessage("Four different founts
 are available ... ",146,0,0)
 150 PROCmessage("Tall text ...",149,1,
0)
```



```
 160 PROCmessage("Giant text ...",147,1
,1)
 170 PROCmessage("Double width text ...
",151,0,1)
 180 PROCmessage("and Normal size text
... ",146,0,0)
 190 PROCmessage("User defined characte
rs may also be employed .. ",150,0,1)
 200 PROCdefine
```

```
  210 PROCmessage(A$,150,0,1)
  220 PROCmessage("There is a choice of
colour ... ",149,0,0)
  230 FOR colour%=145 TO 151
  240 PROCmessage("Colour "+STR$(colour%
)+" ...",colour%,0,0)
  250 NEXT colour%
  260 UNTIL 0
  270 END
  280 :
 1000 DEF PROCinitialise
 1010 VDU 23,1,0;0;0;0;
 1020 osbyte=&FFF4:osword=&FFF1
 1030 params=&8E
 1040 PROCassemble
 1050 ENDPROC
 1060 :
 1070 DEF PROCmessage(message$,col,H,W)
 1080 ?&75=col:?params=H:?(params+1)=W
 1090 $&B00=message$:CALL gianttext
 1100 ENDPROC
 1110 :
 1120 DEF PROCdefine
 1130 VDU23,224,60,126,219,255,126,60,36
,66
 1140 VDU23,225,0,31,17,17,255,191,255,4
8
 1150 VDU23,226,255,36,36,255,255,255,25
5,0
 1160 VDU23,227,248,140,134,255,255,255,
255,24
 1170 B$=CHR$224+" "
 1180 C$=CHR$225:D$=CHR$226:E$=CHR$227
 1190 A$=STRING$(7,B$)+C$+D$+E$+" "+B$+C
$+D$+D$+E$+" "+B$+C$+D$+D$+D$+E$+" "+STR
ING$(7,B$)+" ... "
 1200 ENDPROC
 1210 :
 1220 DEF PROCassemble
 1230 FOR pass%=0 TO 2 STEP 2
 1240 P%=&900
 1250 [ OPT pass%
 1260 .gianttext
 1270 LDY #&00:LDX #&18
 1280 LDA #&00:STA &70:LDA #&7C:STA &71
 1290 .setup
 1300 LDA #&9A:STA (&70),Y:INY
 1310 LDA &75:STA (&70),Y:DEY
 1320 JSR add40:DEX:BNE setup
 1330 :
 1340 .scrollingtext:LDX #&00
 1350 .loadchar:STX &72
 1360 LDA &B00,X:CMP #&0D
 1370 BNE printcharacter
 1380 RTS
 1390 :
 1400 .printcharacter:STA &7F:LDA #&0A
 1410 LDX #&7F:LDY #&00:JSR osword
 1420 LDX #&08
 1430 .decoderow:STX &73:LDA #&27
 1440 STA &70:LDA #&7C:STA &71:LDX #&00
 1450 .decodecol:LDA #&FF:ASL &80,X
 1460 BCS fill:LDA #&00
 1470 .fill:LDY #&00
 1480 STA &74 :STA (&70),Y:JSR add40
 1490 LDA params:BEQ hskip
 1500 LDA&74:STA(&70),Y:JSRadd40
 1510 .hskip
 1520 LDA &74 :STA (&70),Y:JSR add40
 1530 INX:CPX #&08:BNE decodecol
 1540 JSR scrollcharacters
 1550 LDA params+1:BEQ wskip
 1560 JSR scrollcharacters
 1570 .wskip:LDX &73:DEX
 1580 BNE decoderow
 1590 LDX &72:INX:BNE loadchar
 1600 :
 1610 .scrollcharacters
 1620 LDA #&13:JSR osbyte
 1630 LDA #&00:STA &70:LDA #&7C:STA &71
 1640 LDX #&18
 1650 .scroll8rows:LDY #&02
 1660 .scrollarow
 1670 INY:LDA (&70),Y
 1680 DEY:STA (&70),Y
 1690 INY:CPY #&27
 1700 BNE scrollarow
 1710 JSR add40:DEX
 1720 BNE scroll8rows
 1730 RTS
 1740 :
 1750 .add40
 1760 CLC:LDA &70:ADC #&28:STA &70
 1770 LDA &71:ADC #&00:STA &71:RTS
 1780 :
 1790 ]:NEXT pass%
 1800 ENDPROC
 1810 MODE7
 1820 REPORT:PRINT" at line ";ERL:END
```

POINTS ARISING POINTS ARISING POINTS ARISING POINTS A

WORLD BY NIGHT AND DAY (BEEBUG Vol.6 No.2)

A production error resulted in a line of data being omitted from the magazine listing (the magazine cassette/disc is unaffected). To correct this proceed as follows:

Delete line 3440 (incorrect), then renumber lines 3380-3430 as lines 3390-3440, and add a new line 3380:

```
3380 DATA -90,-66.5,-23.5,0,23.5,66.5,90,-9999
```

5 Acorn Launch Archimedes

SOFTWARE

The operating system (known internally as Arthur during development) is an extended version of the operating system used on previous BBC micros. It is much more modular in structure, incorporating a font manager, WIMPs manager, sprite manager and sound system. It is also open to the user (and third party software suppliers) to write further relocatable modules. Once loaded such modules appear as an integral part of the operating system as far as the user is concerned.

A new ARM Basic is supplied, written by Roger Wilson and known as Basic V. This incorporates many new features making the already excellent BBC Basic better than ever. The opportunity has also been taken to correct bugs in earlier versions. Basic V incorporates a full ARM assembler in just the same way that previous versions have incorporated a 6502 assembler.

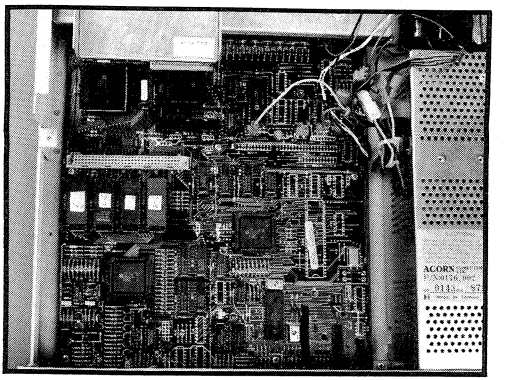
A Welcome Disc will also be provided with a variety of additional applications and demonstration software. Of particular importance will be the multi-coloured Desktop and an excellent music system front end.

Two manuals will be supplied with the new systems, a thin 'get-you-started' Welcome Guide and a completely new Basic User Guide. Detailed reference manuals will be available at additional cost. The manuals are spiral-bound with the Archimedes logo on the front cover.

FIRST IMPRESSIONS

I have to say that I find Archimedes, even at this very early stage, a tremendously thrilling and exciting prospect. The leap from 8 bit to 32 bit architecture coupled with RISC technology combine to produce a machine of the highest quality, which will surely be highly prized by those enthusiasts lucky enough to get hold of one in the coming months.

Whether Acorn have got the price right only time will tell. No doubt there will be many critics, as with the Master series, who will say that the price is just too high, but look at how many Masters have been sold (over 100,000)! Archimedes certainly ups the basic cost of a BBC micro by about 50% (compared with a Master 128 and single 3.5" disc drive).



Inside Archimedes

Home users have to add VAT to the quoted prices, and the systems do then begin to look quite expensive.

The key to the success of Archimedes must surely lie in the availability of a wide range of applications software. A machine of such power (and complexity) will otherwise demand more technical skill than many potential users are prepared or able to provide. And this time Acorn have indicated that much greater reliance is being placed on third party software houses. Whether the continued use of Acorn's own operating system is a wise decision will be much debated. My initial thoughts are that if only Acorn had supplied a really good MS-DOS emulator then Archimedes would be an undoubted winner (maybe it's still not too late). However, Acorn has chosen the education market as its primary target for this machine, and must believe that the right decision has been made.

Whatever is said of Archimedes in the following months, and regardless of its ultimate success or failure, the machine itself is surely a landmark in the history of micro-computing. Much will depend upon it also being a commercial success.

WIN ARCHIMEDES

See page 57 for details of our exciting competition in which you have a chance to win one of Acorn's new super micros.

ACORN LAUNCH ARCHIMEDES

FIRMWARE AND SOFTWARE

In this second article on Archimedes, Lee Calcraft reports on the machine's firmware and bundled software.

Without good software, the most expensive micro in the world (or indeed the fastest) is but nothing. So what do Acorn provide the Archimedian purchaser? The answer is simple: an 800K Welcome disc, plus operating system and Basic in ROM - in fact in four ROMs (you need four to make up the 32 bit word width).

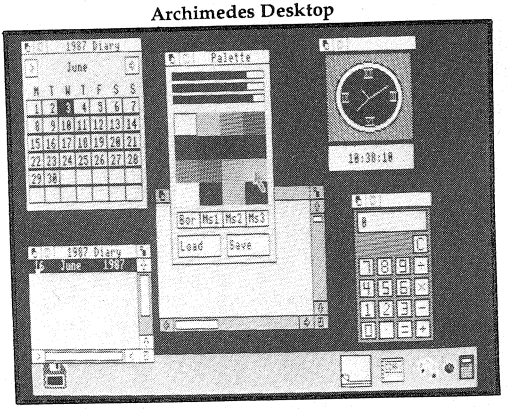
BUNDLED SOFTWARE

At the time of writing (11 June), and with the launch just 5 days away, the content of the Welcome disc is still in a state of flux. The vexed questions of what will be finished in time, and what is there room for on the disc are not fully resolved. But we give Acorn's best estimate in Table 1. Of these items, one of the most impressive in many respects is the suite of desk-top utilities. These encompass the usual disc front end, memo pad, clock, diary and calculator, and in conception they owe much to the Macintosh and to Gem. The mouse is used to select icons and pull-down windows, so that the user can control the computer just by clicking away at the mouse, making the keyboard virtually redundant. The whole palette for the desk-top can be adjusted with almost infinite gradations of colour and shade, and the size and position of any window created by the user is completely under mouse control. This software is exceptionally easy to use and is extremely impressive.

Acorn Desktop
Music Package
Painting Package
Basic Editor
Font Designer
Sprite Definer
6502 Emulator
RAM Basic
Machine Code Debugger
Machine and ADFS Tutorials

Table 1
Probable Contents of the Welcome Disc

Archimedes Desktop



What makes it the more remarkable is that the launch version of this software, which we used, was all written in Basic. It is nevertheless VERY fast, and windows can be opened and dragged at 6502 machine code speed. By the time that production is ramped up in September, this software should be in machine code, and hopefully this will allow the user to engage the disc front end without disturbing his own resident software. As currently supplied, the desktop software will let you move around the ADFS directories with consummate ease, but once you have loaded your Basic program or text file, you are stuck in the currently selected directory. The only way to move around subsequently, without disturbing the contents of memory, is to type in stone age commands like

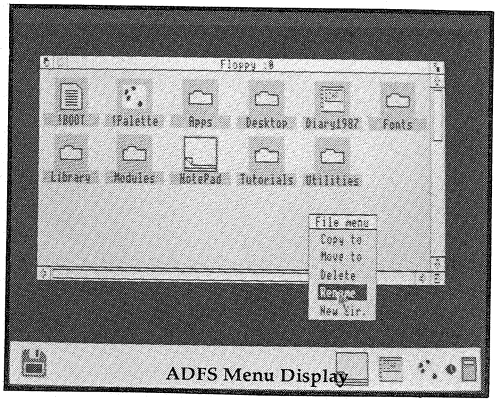
*DIR $.JUNEWORK.TEXT

Conspicuous by its absence from the package is a word processor. A300 series purchasers currently get no bundled word processor or text editor. A Basic line editor is, however, supplied on the Welcome disc, and Acorn have plans to issue a word processor free (yes free!) to all RISC machine purchasers within a few months - just send back the machine registration card to get on the books. We understand that a general purpose editor like that supplied with the Master will be marketed for the machine shortly.

6502 EMULATOR

In a valiant effort to maintain maximum compatibility with the Beeb, Acorn are putting a 6502 emulator on the Welcome disc. This will allow the Archimedes to run 6502 machine code programs - providing that they are very very clean. The

emulator will run View, apparently, but not Wordwise, because to achieve its speed Wordwise writes directly to the screen. In many respects the Emulator software makes the system behave as if it were a RISC machine with a 6502 co-processor. It is claimed at Acorn that the Emulator will run 6502 code at about the speed of a model B - which is pretty impressive. Moreover, the Emulator will even run Basic IV. That is to say it treats Basic IV, quite rightly, like a piece of 6502 machine code. You can then run Basic programs on Basic IV, which is itself running on the Emulator.



ADFS Menu Display

EIGHT CHANNEL SYNTHESIZER

When the VIDC (video controller) ARM chip was being designed, there was enough free silicon to incorporate on it a fully analogue eight channel audio system with stereo capability. The Welcome disc contains a music package to show off the massive potential of this system. In its pre-launch form this was again written in Basic, and had a repertoire of just one piece: Greig's Peer Gynt. But this was more than enough to demonstrate some of the potential of the hardware. The Welcome music package again makes full use of the mouse and the machine's windowing capabilities to allow you to drag notes and clefs on to the musical stave to make up the score, with the greatest ease. Even when playing through its small internal speaker the power, range and potential of the sound system is fully apparent.

ARM BASIC

As we said earlier, the two A300 series machines carry the BBC name, and are both supplied with BBC Basic. The Basic interpreter, like that of the Beeb, was written by Roger Wilson, one of the RISC "pioneers", and takes full advantage of the speed of the RISC processor. This entirely new implementation has been christened Basic V by Acorn, and contains a number of extremely useful enhancements, including limited line editing, limited syntax checking on entry, more line numbers, the inclusion of a set of matrix operators, and a completely rewritten string storage handler. Moreover, it will even translate OSBYTE and other legal operating system calls to ensure optimum compatibility with existing BBC Basic programs. But this is only the start.

Bench-mark	Modl B Bas 2	Mastr Bas 4	Archimedes ROM	Archimedes RAM
Intmath	2.6	2.5	0.35	0.26
Realmath	5.8	4.3	0.38	0.28
Triglog	80.6	43.0	1.41	1.02
Textscrn	13.7	14.2	4.4	4.2
Grafscrn	21.5	22.0	6.9	6.5

Table 2
PCW Benchmarks
(Archimedes timed by Acorn)
ROM - Basic in ROM
RAM - Basic loaded into RAM. This is faster because of RAM paging.

ARM Basic has two entirely new control structures: a WHILE loop, and a CASE statement. The former behaves similarly to the REPEAT loop, except that the condition test is carried out BEFORE the loop is executed. This contrasts with the REPEAT loop, which will be executed once even if the condition is never satisfied. The CASE statement provides a quite complex control structure using the sequence of keywords:

CASE..OF..WHEN..OTHERWISE..ENDCASE

It can be used to good effect in certain quite specific applications, such as a menu which requires different responses to a set of possible key-presses. A block structured IF statement is also implemented, allowing the IF-THEN-ELSE construct to extend over a number of lines through the uae of the keyword ENDIF.

MODES, COLOURS AND SOUNDS

A300 series machines have 21 possible screen modes, none of which are shadow modes. The first 8 are similar to those on the standard model B, except that modes 0 to 6 all utilise 20K of screen memory,

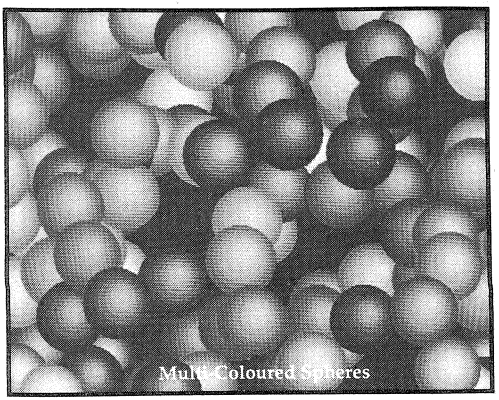
while mode 7 uses a frightening 80K! This is because the teletext mode is software-implemented. The new modes are given in the accompanying table, and you can see that they use up to 160K of RAM - which is already quite a gulp if you only have 512K. If you are not short of RAM, you will find mode 15 particularly useful. This gives 80 column text, 640x256 resolution, and 256 colours on screen at the same time.

8	80x32	640x256	4	40K
9	40x32	320x256	16	40K
10	20x32	160x256	256	40K
11	80x25	Text	4	40K
12	80x32	640x256	16	80K
13	40x32	320x256	256	80K
14	80x25	Text	16	80K
15	80x32	640x256	256	160K
16	132x32	Text	16	132K
17	132x25	Text	16	132K
18	80x32	640x512	2	40K
19	80x32	640x512	4	80K
20	80x32	640x512	16	160K

Table 3
The New Screen Modes

Modes 0 to 7 are the same as previously except that modes 0 to 6 all use 20K, and mode 7 uses 80K. Modes 18 to 20 may only be used on special multi-sync monitors.

As with all modes but 7, the available colours may be selected from a palette of 4096 colours, though selection from this large palette is only possible in groups



Multi-Coloured Spheres

of 16. Modes 18-20 require a special multi-sync monitor, and will be unusable on normal monitors. And incidentally, the output from the Archimedes machines is ANALOGUE RGB. The Beeb's output is TTL or digital RGB, and the two are very different. Existing Beeb owners will not be able to use their standard Microvitecs and the like with the new machines, though the Philips monitors bundled with the Compact do have a suitable input socket, and there is no problem for mono users - except that the Archimedes mono video socket is a phono type rather than the better quality BNC on the Beeb.

The way in which the enormous variety of colours and tints are selected seems a little involved. Up to 64 different logical colours may be defined with relative ease using the Archimedes extended COLOUR command:

COLOUR n,r,g,b

This assigns r degrees of red, g degrees of green, and b degrees of blue to logical colour n. The variables r, g, and b can take integer values between 0 and 15, giving 16x16x16 possible hues, or 4096 in all. If you require more than 64 different physical colours on screen, it appears that you must resort to a TINT command to give what the manual refers to as "four subtle variations to each colour".

SOUND EXTENSIONS

The Archimedes has 8 sound channels, and these work in an entirely different way to those on the Beeb, each being able to completely simulate any given instrument (in theory at least). On the Beeb, you need to combine all three tone channels to build up a waveform of any credibility. Acorn have retained the SOUND command, with its 16 levels of output, but there is now an optional logarithmic mode, which will make volume changes much smoother. There is also a much more precise definition of pitch optionally available. This is now capable of 4096 different pitch levels within each octave.

The ENVELOPE command used in the Beeb now has no effect whatsoever on the Archimedes. Instead the user is provided with a set of commands: VOICES, TEMPO, BEAT, and STEREO. Even so, there is no real equivalent to the ENVELOPE command, and no simple way of defining the timbre of each channel. The Welcome disc contains a number of modules giving different types of sound such as percussion, strings etc, and a skeleton module is provided for user customising. But it looks as if the user

INTELLIGENT MODEMS

Peter Rochford, BEEBUG's communications expert, reports on the latest modems now available to all avid comms enthusiasts, and everyone else as well.

In the last Comms Spot we took a detailed look at the subject of intelligent modems and what makes them so much more advanced than their predecessors in terms of facilities and ease-of-use.

This month we review six of the latest intelligent modems from three of the largest modem manufacturers in the UK. It is advisable to have read the last Comms Spot (BEEBUG Vol.6 No.2) before reading this review, unless you already have experience or knowledge of intelligent modems. I have not described facilities common to all these modems, having already done so in the previous article.

Three of the modems reviewed are at the lower end of the price spectrum, whilst the other three are definitely aimed for the moment at business users, or, comms enthusiasts with plenty of disposable income. You may well question the point of reviewing three modems which can cost over £500 each in a magazine that caters mostly for home computer users. Well, the reason is this. These modems can operate at 2400/2400 baud and soon Prestel and other BT information services will be running on a network that enables communication at that speed. When that happens, interest in high speed modems will increase and I feel sure that the prices will fall markedly. In the case of the Miracle modems and the Pace Series Four, they can be purchased as V21/V23 initially and then upgraded to V22 or V22bis as finances allow.

GENERAL

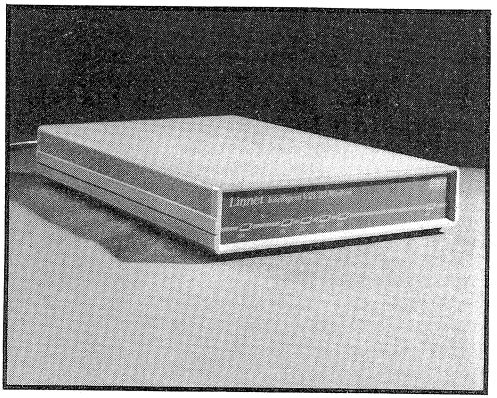
All the modems reviewed here have full BABT approval, and connect to the telephone line via the BT 600 type plug and socket. Rear panels of the all the modems feature a telephone socket and a 25-way 'D' type RS232 socket for connection to the computer.

I should point out to potential purchasers of any of these modems that they do not come with a lead for connection to the computer as standard and you would be best advised to get your dealer to make one up for you.

PAGE LINNET
Price: £163.30 inc VAT (Price includes Commstar II ROM)

This unit is housed in a very slim beige case with a grey front panel that features six LEDs for indicating power, on-line, carrier detect, transmit, receive and auto-answer. The external construction and finish of the Linnet is excellent and it looks a neat and attractive unit.

Apart from the standard connections on the rear panel, there is a DIN socket that connects to the modem's external PSU. The PSU is of the type that plugs into a wall socket via pins moulded into the housing. In use, I found that this ran very warm after a while, but did not cause any problems during the review.



The only other feature on the Linnet's rear panel is a small push switch. This is for resetting the modem's own software to the initial power-on state, and is very handy for aborting any operation quickly without having to send an instruction from the computer.

Documentation consists of a 56-page manual which is extremely well-written

and laid out and which provides plenty of information to satisfy both beginner and expert alike. It is a pity that more user manuals are not written in this fashion!

The Linnet responds to the standard Hayes AT command set (see last month) and has the usual 'S' registers too. However, Pace have taken things a stage further by incorporating extra AT commands, some 11 in all, to enhance the standard set. Also, they have included some new S registers that provide greater flexibility in configuring the modem.

Unusual too, are the 'help' screens available in the Linnet's software. These are called up from the modem for viewing on the terminal screen. This allows displays of the AT command set, and the S registers and their programmed values.

A further refinement is the Linnet's ability to check the status of the telephone line and report back to the terminal. For example, it can detect whether there is a dial tone and if the called number is engaged. If the number is engaged, the modem will make several attempts to re-dial the number after preset periods.

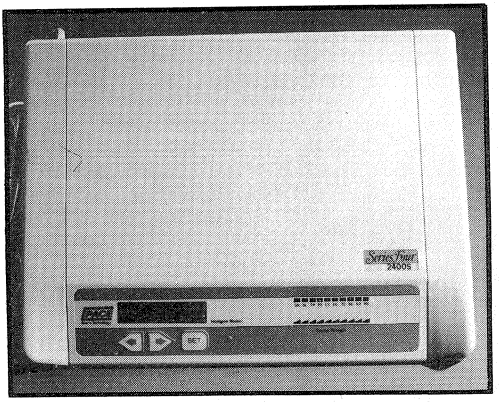
VERDICT

It did not take long using the Linnet to become very impressed with it in so many ways. The software in the modem is excellent and makes it a delight to use, with extra features that really do make all the difference. I find it difficult to fault the Linnet at all. An excellent modem from Pace, at a sensible price. Very highly recommended for those who only need V21/V23 operation.

PACE SERIES FOUR
Price: £280.00 inc VAT (V21/V23)
: £458.85 inc VAT (V21/V23/V22)
: £573.85 inc VAT (V21/V23/V22/ V22bis)

This is a most remarkable looking modem. It is housed in a beige wedge-shaped case with the display and controls at the front edge. There are 10 LEDs for monitoring RS232 status and beneath them, 10 more LEDs that monitor the strength of the incoming carrier. To the left of these is a 32 character LCD display and below that, three pressure sensitive switches. If you like hi-fi with flashy front panels you will love the Series Four!

The LCD display gives a constant date and time readout, details about call progress, and, in combination with the switches, allows many of the modem's functions to be controlled independently of the terminal it is connected to. At the rear of the modem are the standard connections plus a DIN socket for the connection of the external PSU. The PSU is the same as used for the Linnet, and like the Linnet, the Series Four also has a system reset switch. Alongside the RS232 socket is a printer port that is Centronics compatible and allows the modem to receive messages which are time and date stamped, without being connected to a computer or terminal.



The documentation consists of an A5 loose leaf binder that is split into several sections. Again, as with the Linnet, Pace have produced a very readable and informative set of user instructions.

In use the Series Four is much like the Pace Linnet when operated via the terminal. The software is virtually identical and my comments in the Linnet review equally apply here too. The Series Four though does have extra features over the Linnet, such as tone dialling and the ability to determine whether tone or pulse dialling is appropriate on the line it is connected to. And, of course, it operates at much higher baud rates which accounts for the much higher price tag.

VERDICT

I started off by saying that this was a remarkable looking modem. Well, it is a remarkable modem in all respects. The

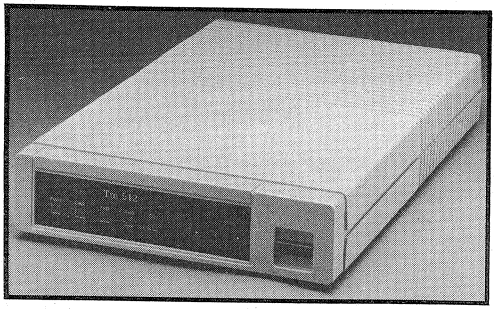
styling leaves me cold I must admit, and to my mind is totally impractical as it does not allow stacking.

But styling aside, technically, this modem is what you might call 'Rolls Royce' or 'state-of-the-art'. It is superb to use and I have to admit to being very sad when I had to return the unit after the review.

If money is no object then certainly the Series Four deserves serious consideration. Hopefully the price may drop in the future but until then, those on a limited budget like myself, can only dream.

TANDATA TM512
Price: £293.25 inc VAT

Tandata may not be a name quite as familiar to some home computer users as, say, Pace or Miracle when it comes to modems. However, they have achieved great success in the business market with their range of modems and viewdata terminals.



The TM512 is their medium-priced Hayes compatible modem that comes housed in a brown plastic case with a dark brown front panel adorned with five LEDs and a power switch. The LEDs monitor power, on-line, transmit data, receive data and carrier detect.

At the rear of the unit are the standard connections plus a six pin DIN socket which may be used for RS232 connection instead of the usual 25-way connector. The modem has an internal power supply, and so comes with a captive mains lead but without an attached mains plug.

Documentation consists of a comprehensive manual. Although in the main well-written, this does not cater so well for the needs of the newcomer to comms.

The TM512 responds to the standard Hayes AT set but can also be controlled using the V25bis command set. The modem features a number store holding up to eight numbers. These can be auto-dialled like those on the other modems in the review, but unlike say the Pace and Miracle modems, the TM512 store can hold not only the number to be dialled but also IDs and passwords to be given up when the host sends its ENQ.

An important feature of this modem is its inbuilt EPAD error correction facility. Those who use PSS/Multistream EPAD service can get error-free communication and operate in a 1200 baud pseudo full-duplex mode using this facility.

VERDICT

I found the Tandata easy to set up and use, and there were no problems with the hardware or software during the review. The extra facilities of the V25 command set and inbuilt ID store do not appeal to me. The EPAD error correction may well be important to those who use PSS, otherwise you are paying for a redundant feature.

I cannot help but say, that I feel this modem is rather overpriced. Unless you need the V25 or EPAD, the Pace Linnet or the Miracle WS4000 are cheaper and perform every bit as well, with the Pace offering far better controlling software.

Our modem reviews will be concluded next month when a detailed summary table covering all the modems reviewed will be published.

Suppliers

Pace Micro Technology,
Allerton Road,
Bradford, West Yorkshire BD15 7AG.
Tel. (0274) 488211

Tandata,
Albert Road North, Malvern,
Worcs. WR14 2TL.
Tel. (0684) 892421

MODE 7 Screen Editor

Ashley Kitson's comprehensive mode 7 screen editor will provide all the facilities that are needed to edit mode 7 screens, whether as a title page to a game, a mailbox message for Micronet, or whatever takes your fancy.

Often the most tedious and difficult process involved in writing a new program is the creation and coding of suitable screen designs. Using a mode 7 editor, however, you can avoid all this. Just create your menu screen, or whatever, using an editor. Then use *LOAD from within your Basic program to display your design on screen. What could be simpler?

The accompanying mode 7 editor should prove ideal for the purpose, as well as for the more obvious mode 7 applications in comms. It is written in machine code, giving both speed and ease of use, and is intended for use with a disc system. For those wishing to put it into sideways RAM or EPROM, it generally conforms to the requirements of the sideways RAM utility published in BEEBUG Vol.4 No.6 page 37.

ENTERING THE LISTING

Type in the program, and save it away. When it is run you should see the following lines printed on the screen once the assembler listing has scrolled past:

```
Length of code = &46Ø
To save code, copy the following:
*SAVE MCEDIT 5ØØØ +46Ø 5ØØØ
```

Please note the value given for the length of code (&46Ø hex). If you get the same then you can be fairly certain that you have typed in the listing correctly. If the figures don't match, you can be sure that you have made a mistake somewhere.

If all is well, copy the last line printed by the program (using the cursor editing keys) to save the machine code version of the program to disc. This can now be run by typing:

*MCEDIT or *RUN MCEDIT

USING THE PROGRAM

Cut out the function key strip on page 36 and place it on your keyboard. Then type *RUN MCEDIT or *MCEDIT to load and run the machine code. The screen should now clear leaving the cursor at the centre. If all is well, you should find that typing at the keyboard will put characters on the screen. The cursor keys work in the expected fashion, moving the cursor around the screen, while, in conjunction with the Shift key, they take you to the edges of the screen. The Tab key will advance the cursor 5 characters to the right.

All of the mode 7 teletext control characters are available by using the function keys by themselves or in combination with the Shift or Ctrl keys as indicated on the keystrip. For an overview of the use of teletext graphics characters you are referred to the BBC Micro User Guide chapter 28, and to its appendix.

Graphics characters can be input, using the editor, by first selecting the appropriate graphics colour code (Shift-fØ - Shift-f6), then the keyboard character which will generate the required shape. The User Guide appendix is useful here in providing a table of available graphics characters and their corresponding keys. Additionally Ctrl-f8 will give the solid block character, not normally available from the keyboard. To create double height characters you will need to insert the double height code (Ctrl-f5) to the left of the screen on both lines to be used, then type in the required character strings, again on both lines.

Ctrl-f9 allows you to enter operating system commands. If your command produces more information than can be held in the window, you can stop the scrolling by pressing the Shift and Ctrl keys together. Beware that any command which uses main memory may corrupt your program, so avoid *COPY, *COMPACT and the like. Also, any command which cannot be carried out by the operating system will terminate the program (see later). Pressing any key after the successful completion of the command will restore your screen.

Ctrl-S allows you to save your screen to or disc. The screen will clear and a catalogue of the currently installed disc will be displayed on screen. Just enter

the filename (seven characters maximum), and press Return.

Ctrl-L allows you to load a previously created screen from disc. As with the save facility, the disc catalogue is presented, and you may use the cursor keys to copy a filename, again with the same proviso about name lengths. In the event that the filename cannot be found, a warning will be given.

When you have finished your editing session, and saved away your design, the editor may be exited using Ctrl-Escape. Pressing the Escape key by itself has no effect within the editor, except when in load, save or star command mode. Escape has been handled in this way to avoid accidental use.

WARNING

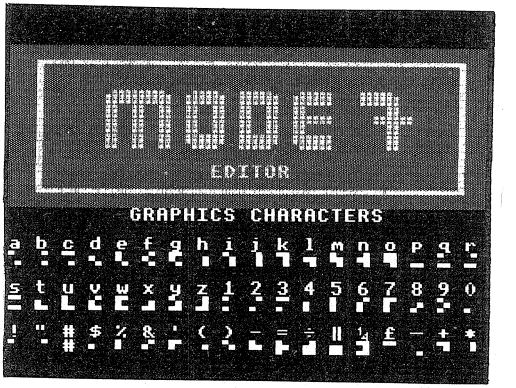
Certain types of error when using the save, load or star command options will cause the editor to exit to Basic with the appropriate error message. If this happens, you may press Break to return you to the editor, with your original screen restored. The program achieves this by saving the current screen to another area of memory (&7800-&7BFF) whenever Ctrl-L, Ctrl-S or Ctrl-f9 are pressed.

You can take advantage of this feature to temporarily store intermediate screen versions of a design, perhaps as a backup. Just press Ctrl-L followed by Escape (not Ctrl-Escape) whenever you require a temporary save. To reinstate this temporary screen at any time, just press Break. When you quit the program with Ctrl-Escape, the definition stored in the Break key is cleared. To re-enter the editor and restore any temporarily saved screen in this case, use:

CALL &5457

from Basic.

```
  10 REM Program Mode 7 Screen Editor
  20 REM Version B1.10(J)
  30 REM Author  A. Kitson
  40 REM Beebug  July 1987
  50 REM Program subject to copyright
  60 :
 100 strt%=&5000
 110 FORI%=0 TO 3 STEP 3
 120 P%=strt%
 130 PROCassemble
 140 NEXT
 150 PRINT
```




```
 160 PRINT"Length of code = &";~P%-strt
%
 170 PRINT'"To save code, copy the foll
owing:"
 180 PRINT'" *SAVE MCEDIT "+STR$~(strt%
)+" +"+STR$~(P%-strt%)+" "+STR$~(com%)
 190 END
 200 :
1000 DEFPROCassemble
1010 [OPTI%
1020 .com%:JSRsetkeys:JSRmode7set
1030 .reen2:JSRscreenwrite:JSRclearkeys
1040 LDA#&1A:JSR&FFEE:LDA#&1F:JSR&FFEE
1050 LDA#0:JSR&FFEE:LDA#&18:JSR&FFEE
1060 RTS
```

MEMORY USAGE

&70	- &8F	General workspace
&900	- &907	Filename
&908	-	OS command
&A00	- &A11	Parameter blocks
&7800	- &7BFF	Temp screen storage

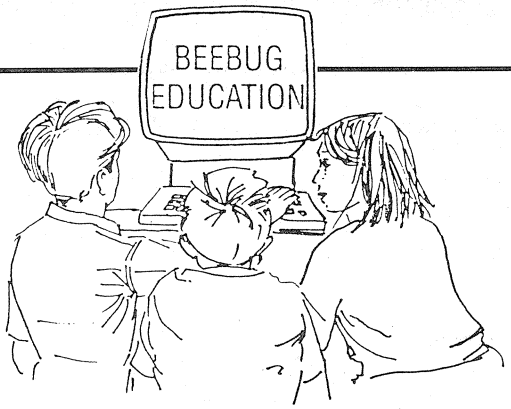
```
1070 :
1080 OPT FNdata
1090 OPT FNfastmove
1100 OPT FNwrtstring
1110 OPT FNrdstring
1120 OPT FNvdu
1130 :
1140 .mode7set:LDX#mode7 MOD256
1150 LDY#mode7 DIV256:LDA#12:JSRvdu:RTS
1160 :
1170 .getfilename
1180 LDX# text MOD256
1190 LDY# text DIV256:JSRwrtstring
1200 LDA#4:LDX#0:JSR&FFF4
1210 LDA#229:LDX#0:LDY#0:JSR&FFF4
1220 LDX#0:LDY#9:LDA#7
1230 JSRreadstring:PHA:LDA#229:LDX#255
```

```
1240 LDY#0:JSR&FFF4:PLA:RTS
1250 :
1260 .setkeys
1270 LDA#229:LDX#255:LDY#0:JSR&FFF4
1280 LDA#225:LDX#129:LDY#0:JSR&FFF4
1290 LDA#226:LDX#145:LDY#0:JSR&FFF4
1300 LDA#227:LDX#156:LDY#0:JSR&FFF4
1310 LDA#4:LDX#1:LDY#0:JSR&FFF4
1320 LDX#k10 MOD256:LDY#k10 DIV256
1330 JSR&FFF7:RTS
1340 :
1350 .clearkeys
1360 LDA#229:LDX#0:LDY#0:JSR&FFF4
1370 LDA#225:LDX#1:LDY#0:JSR&FFF4
1380 LDA#226:LDX#0:LDY#0:JSR&FFF4
1390 LDA#227:LDX#0:LDY#0:JSR&FFF4
1400 LDA#4:LDX#0:LDY#0:JSR&FFF4
1410 LDX#k10clear MOD256
1420 LDY#k10clear DIV256:JSR&FFF7:RTS
1430 :
1440 .savescrn:JSRdownscreen
1450 LDX#savetext MOD256
1460 LDY#savetext DIV256:JSRwrtstring
1470 JSRcatdisc:JSRgetfilename:BNEsav
1480 JSRMakeSaveLoadBlock
1490 JSRsavepara:LDA#0:JSRlosascreen
1500 .sav:LDA#4:LDX#1:JSR&FFF4
1510 JSRupscreen:JSRsetp:RTS
1520 :
1530 .loadscreen:JSRdownscreen
1540 LDX#loadtext MOD256
1550 LDY#loadtext DIV256
1560 JSRwrtstring:JSRcatdisc
1570 JSRgetfilename:BNEloadsl:LDA#0
1580 STA&A00:LDA#9:STA&A01:LDX#0:LDY#&A
1590 LDA#5:JSR&FFDD:CMP#0:BNEloads
1600 LDX#fnf MOD256:LDY#fnf DIV256
1610 JSRfilerror:JMPloadsl
1620 .loads:JSRMakeSaveLoadBlock
1630 JSRloadpara:LDA#&FF:JSRlosascreen
1640 .loadsl:LDA#4:LDX#1:JSR&FFF4
1650 JSRupscreen:JSRsetp:RTS
1660 :
1670 .losascreen:LDX#0:LDY#&A
1680 JSR&FFDD:RTS
1690 :
1700 .catdisc:LDA#ASC".":STA&908
1710 LDA#&D:STA&909:LDX#8:LDY#9
1720 JSR&FFF7:RTS
1730 :
1740 .MakeSaveLoadBlock:LDX#&11:LDA#0
1750 .ssl:STA&A00,X:DEX:BPLssl
1760 LDA#&FF:STA&A04:STA&A05:STA&A0C
1770 STA&A0D:STA&A10:STA&A11
1780 LDA#9:STA&A01:RTS
1790 :
1800 .savepara:LDA#&7C:STA&A03:STA&A0F
1810 LDA#&78:STA&A0B:RTS
1820 :
1830 .loadpara:LDA#&78:STA&A03
1840 LDA#0:STA&A06:RTS
1850 :
1860 .filerror:JSRwrtstring
1870 JSR&FFE0:RTS
1880 :
1890 .downscreen:LDA#&7C:STA&71
1900 LDA#&78:STA&75:JSRcommonscreen
1910 JSRfastmove:LDA#&C:JSR&FFEE:RTS
1920 :
1930 .upscreen:LDA#&C:JSR&FFEE
1940 LDA#&78:STA&71:LDA#&7C:STA&75
1950 JSRcommonscreen:JSRfastmove:RTS
1960 :
1970 .commonscreen:LDA#0:STA&70
1980 STA&72:STA&74:LDA#4:STA&73:RTS
1990 :
2000 .oscli:JSRdownscreen
2010 LDX#wintext MOD256
2020 LDY#wintext DIV256
2030 JSRwrtstring:LDA#229:LDX#0
2040 LDY#0:JSR&FFF4:LDX#8:LDY#9
2050 LDA#&FF:JSRreadstring:PHA:TYA:PHA
2060 LDA#229:LDX#255:LDY#0:JSR&FFF4
2070 PLA:TAY:PLA:BEQosl
2080 JMPclosewindow
2090 .osl:LDA#&D:STA&908,Y
2100 LDX#8:LDY#9:JSR&FFF7:JSR&FFE0
2110 .closewindow:JSRupscreen:JSRsetp
2120 JMPscwr
2130 :
2140 .screenwrite:LDA#20:STA&80
2150 LDA#12:STA&81:JMPmove
2160 .scwr:LDA#&FF:STA&8E:JSR&FFE0:
2170 BCCsc:JMPscend:.sc:PHA
2180 JSRshiftcheck:PLA
2190 CMP#&A0:BNEsc0:LDA#&8C:JMPsc7
2200 .sc0:CMP#&A1:BNEsc1:LDA#&8D:JMPsc7
2210 .sc1:CMP#&A2:BNEsc2:LDA#&88:JMPsc7
2220 .sc2:CMP#&A3:BNEsc3:LDA#&89:JMPsc7
2230 .sc3:CMP#13:BNEsc4
2240 INC&81:LDA#0:STA&80:JMPmove
2250 .sc4:CMP#9:BNEsc5::LDA#5:CLC
2260 ADC&80:STA&80:JMPmove:.sc5:CMP#&1B
2270 BNElload:PHA:LDA#&81:LDY#&FF
2280 LDX#&FE:JSR&FFF4:PLA:CPX#&FF
2290 BEQscend
2300 :
2310 .lload:CMP#&C
2320 BNEsc5a:JSRloadscreen:JMPscwr
2330 .sc5a:CMP#&13:BNEsc5b:JSRsavescrn
2340 JMPscwr:.sc5b:CMP#32:BCCscwr
2350 CMP#&88:BNEsc5c:LDA#0:STA&8E
2360 JMPleft:.sc5c:CMP#&89:BNEsc5d
2370 LDA#0:STA&8E:JMPright:.sc5d
2380 CMP#&8A:BEQdown:CMP#&8B
2390 BEQup:CMP#&A5:BNEsc6:JMPoscli
2400 .sc6:CMP#&A4:BNEsc7:LDA#&FF:.sc7
2410 CMP#&7F:BNEsc9:LDA#0:CMP&80:BNEsc8
2420 CMP&81:BNEsc8:JMPscwr:.sc8:LDA#&7F
2430 JSR&FFEE:JMPleft:.sc9:PHA:LDA&80
```

```
2440 CMP#39:BNEscl0:LDA&81:CMP#24
2450 BNEscl0:PLA:STA&7FE7:JMPscwr
2460 .scl0:PLA:JSR&FFEE:JMPright
2470 .scend:RTS
2480 :
2490 .left:DEC&80:LDA&8E:BNEleftl
2500 LDA&8F:BEQmove:LDA#0:STA&80
2510 .leftl:JMPmove:.right:INC&80
2520 LDA&8E:BNErightl:LDA&8F:BEQmove
2530 LDA#39:STA&80:.rightl:JMPmove
2540 .down:INC&81:LDA&8F:BEQmove
2550 LDA#24:STA&81:JMPmove
2560 .up:DEC&81:LDA&8F:BEQmove
2570 LDA#0:STA&81:.move
2580 JSRcheckmove:.moveok:JSRsetp
2590 JMPscwr:.checkmove:LDA&80
2600 BPLmovel:LDA#39:STA&80:.movel
2610 CMP#40:BCCmove2:LDA#0:STA&80
2620 INC&81:.move2:LDA&81:BPLmove3
2630 LDA#24:STA&81:RTS:.move3
2640 CMP#25:BCCmove4:LDA#0
2650 STA&81:.move4:RTS
2660 :
2670 .setp:LDA#&1F:JSR&FFEE:LDA&80
2680 JSR&FFEE:LDA&81:JSR&FFEE:RTS
2690 :
2700 .shiftcheck:LDX#&FF:LDY#&FF
2710 LDA#&81:JSR&FFF4
2720 STY&8F:TXA:AND&8F:STA&8F:RTS
2730 :
2740 .reenter:JSRsetkeys:JSRupscreen
2750 JMPreen2
2760 ]:ENDPROC
2770 :
2780 REM ** LIBRARY ROUTINES **
2790 DEFFNfastmove
2800 LOCALstrt%,finish%,new%,fastmove0,
fastover1,fastover2
2810 strt%=&70:length%=&72:new%=&74
2820 fastmove0=P%+2:fastoverl=P%+&C
2830 fastover2=P%+&12
2840 [:OPT I%
2850 .fastmove:LDY#0:.fastmove0
2860 LDA(strt%),Y:STA(new%),Y
2870 INCnew%:BNEfastoverl:INCnew%+1
2880 .fastoverl:INCstrt%:BNEfastover2
2890 INCstrt%+1:.fastover2
2900 DEClength%:BNEfastmove0
2910 DEClength%+1:BNEfastmove0:RTS
2920 ]:=I%
2930 :
2940 DEFFNwrtstring
2950 LOCALB,L%
2960 B=P%+16:L%=P%+6
2970 [OPTI%
2980 .wrtstring:STX&70:STY&71
2990 LDY#0:.L%:LDA(&70),Y:BEQB
3000 JSR&FFE3:INY:BNEL%:.B:RTS
3010 ]:=I%
3020 :
3030 DEFFNrdstring
3040 LOCALreadstringl
3050 readstringl=P%+&26
3060 [OPTI%
3070 .readstring:STX&A00:STY&A01
3080 STA&A02:LDA#32:STA&A03
3090 LDA#122:STA&A04:LDA#0:LDX#0
3100 LDY#&A:JSR&FFF1:BCCreadstringl
3110 LDA#&7E:JSR&FFF4
3120 LDA#&FF:RTS
3130 .readstringl:LDA#0:RTS
3140 ]:=I%
3150 :
3160 DEFFNvdu
3170 LOCALvdul:vdul=P%+5
3180 [OPTI%
3190 .vdu:STX&70:STY&71:TAX:DEX
3200 LDY#0:.vdul:LDA(&70),Y:JSR&FFEE
3210 INY:DEX:BPLvdul:RTS
3220 ]:=I%
3230 :
3240 DEF FNdata
3250 [OPTI%:.text
3260 EQUS"Enter Filename >":BRK
3270 .wintext:EQUS"Enter command *":BRK
3280 .savetext:EQUB&82
3290 EQUS"Save Screen":EQUB13:BRK
3300 .loadtext:EQUB&82
3310 EQUS"Load Screen":EQUB13:BRK
3320 .fnf:EQUB&D:EQUB7:EQUB&85
3330 EQUS"File not found":BRK
3340 .mode7:EQUD&1F0C0716:EQUW&C12
3350 EQUD&27180001C:EQUW&C00
3360 .kl0:EQUS"K.10 CA.&"+STR$~(reenter
)+"|M":EQUB&D
3370 .kl0clear:EQUS"K.10":EQUB&D
3380 ]:=I%
```

13 Personal Address Book

```
3110 NEXT col
3120 NEXT row
3130 R%=R%+1:IF R%>T% T%=R%
3140 PROCshowpage
3150 ENDPROC
3160 :
3170 DEFFNt
3180 IF T%>0 THEN =TRUE
3190 REMT$=FNchoice("No entries in book
",s$,s$,5)
3200 B%=0:R%=0:PROCwl(1):PROCshowpage
3210 =FALSE
3220 :
3230 DEFFNstrip(A$)
3240 A$=A$+s$
3250 REPEAT
3260 A$=LEFT$(A$,LENA$-1)
3270 UNTIL RIGHT$(A$,1)<>s$
3280 =A$
```



In this month's BEEBUG Education, Mark Sealey assesses some of the latest software aimed at the educational market.

This time I have decided to concentrate on reviews of some of the many new items of educational software that are published each month. Each is distinguished in some way by its usefulness to teachers, or by breadth of application.

Product : Miscue Analysis
Supplier : MRH Systems and Software
20 Highfield Road,
Kidderminster,
Worcestershire DY10 2TL.
Price : £15.00

Some ideas are so simple that you wonder why they have not been implemented before. Just such an idea obviously inspired the suite of programs on this disc. Surprisingly little use is made of 'Expert Systems' in education, but here are two programs that mimic expert systems and are based on sound educational principles.

The first 'diagnoses' on-screen the miscues (not mistakes) that children (of all ages and abilities) can make when reading. Firstly, record the miscues as they occur: with the excellent documentation there is a specially prepared sheet for this purpose. Then you are lead through a hierarchical, tree-like description with simple causes and remedies suggested for each type of miscue. These are comprehensive and clear.

Further, a file of the analyses and dates arrived at by this process (on up to 30 children on each data disc) can be stored and/or added to later. This data also contains type-counts for each child, and as such can make the task of class record keeping much easier. Given the increasing importance attached to record keeping, there is now no excuse where reading is concerned!

Printed output is available at all suitable points, as well as easy selection (by menu) of disc surfaces, etc. I have always suspected that programs that make teachers' lives easier are a sure-fire success. This one should be too!

Next a comparison of two Statistics packages:

Product : Statspak
Supplier : Chalksoft
P.O. Box 49,
Spalding,
Lincs PE11 1NZ.
Tel. (0775) 69518
Price : £17.50 (2 discs) inc. VAT
£19.80 (for Compact) inc. VAT

Product : "Scigraf and Regress"
Supplier : John Wiley
Baffins Lane,
Chichester,
Sussex PO19 1UD.
Tel. (0243) 784531
Price : £29.95 inc. VAT

These statistical packages will attract two different sets of users: firstly the teachers of maths and statistics, or indeed any other subject (like geography), where manipulation and presentation of data is relevant; and secondly the professional use by teachers, academics and lecturers for the purpose of educational research.

Age-ranges: Statspak is aimed at those covering 'O', GCSE and 'A' level syllabi, but would be suitable as introductory material for younger users (a pre-'O' level environmental project for example). Scigraf is probably better suited to students at a higher level. The documentation for the latter is superior, fuller, and with more of a tutorial flavour. Potential buyers, however, will want to know which functions each package can handle and how well.

STATSPAK

Chalksoft's package can handle mean, mode, median, variance, standard deviation, standard error of the mean,

coefficients (variation and skewness), frequency distribution, quartiles and interquartile and semi-interquartile ranges on single and multiple sets of numbers. Both regressions of x and y, and Pearson, Spearman and Kendall's Coefficients as well as CHI square. Although the displays are not always easy to read, there is a built-in Epson dump to produce hard copy of the various displays (e.g. frequency polygon, scatter graph or histogram etc).

Although instructions to swap discs at appropriate points are clear, and the saving, retrieving and managing of data on the disc reliable, there are some grumbles (scaling on scatter diagrams for instance, and the impossibility of comparing two distributions simultaneously). A major disappointment lies in the way numbers are entered, a dot disappearing for each one with no indication or trapping of impossibly long sequences of digits. The copy cursor is not trapped, and the displays are generally less than ideal. This points to the need to use this package under close teacher supervision.

SCIGRAF

This is more concerned with plotting than the pure handling of statistics. It handles plotting with linear, as well as logarithmic x and y, normal distribution and log-normal distribution curves. The data on which these graphs are based is held in common databases on disc, and transferring between them in any order is easy. As with Statspak, there are some good examples on disc. Scaling is automatic (with log too), but data can also be plotted to scales of the user's own choice. There is no theoretical limit to the amount of data that can be handled, and it can be displayed as points, lines, broken lines or bar-charts.

The calculation of formulae is automatic and speedy, and for those with an Epson compatible printer there is an in-built dump routine. Additionally, graph plots can be saved as files for further use in other environments. There is an almost exemplary 'technical appendix' which contains sections on trouble-shooting, transfer to ADFS, possible sideways ROM clashes etc. The appendix does give examples of use and adaptation (by Basic programs for editing large data files for instance). It also outlines some of the mathematical theory on which the package is based.

From this it will be obvious that Scigraf is the more specialised package. If you want to display or demonstrate polynomials or correlations, either for their own sake or related to project work, then this is for you. If it's heavier number-crunching, and you can put up with the minor irritations mentioned above, try Statspak.

Product : Muddles'
Supplier : Software Production Associates
PO Box 59,
Leamington Spa,
Warwicks CV31 3QA.
Tel. (0926) 22959
Price : £13.80 inc. VAT

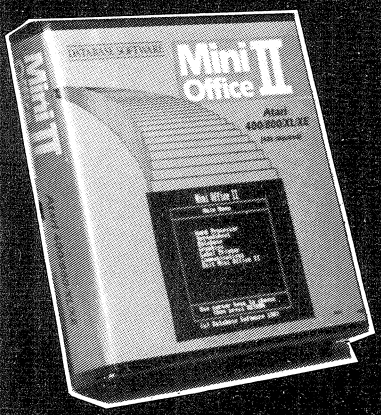
"Muddles is a language development program for all ages. It is suitable for infant and primary classrooms, middle and some lower secondary schools and in special needs departments anywhere. Additionally it could be used in modern language departments in colleges, universities and in adult literacy work." What a claim!

It displays colourful, clear, mode 7 text in short chunks - typically eight or ten words - which is muddled in one or more ways. All the vowels might be missing, the spaces might have been omitted, or the words may appear in reverse or in reverse order, etc.

The aim is then to make sense of the line by entering your version: if you don't get it from one 'muddled' version, another will help you, while any correct letters stay put. Control via space-bar and cursor keys is unusually easy. You can readily alter the level of difficulty too. The way that "Muddles" expects the user to make sense is by meaning, not decoding.

The package thus takes account of theories of reading that now (thankfully) have preference over phonic reading schemes. The whole feeling is right and assumes children are ABLE, enthusiastic people keen to learn. With this package - they will. Worth buying. →54

Mini Office II



6 powerful home and business programs in just one package – at a price that simply can't be matched!

- WORD PROCESSOR: Compose a letter, set the print-out options using embedded commands or menus, use the mail merge facility to produce personalised circulars – and more!
- DATABASE: Build up a versatile card index, use the flexible print out routine, do powerful multi-field sorting, perform all arithmetic functions, link with the word processor – and more!
- LABEL PRINTER: Design the layout of a label with the easy-to-use editor, select label size and sheet format, read in database files, print out in any quantity – and more!
- SPREADSHEET: Prepare budgets or tables, total columns or rows with ease, copy formulae absolutely or relatively, view in either 40 or 80 column modes, recalculate automatically – and more!
- GRAPHICS: Enter data directly or load data from the spreadsheet, produce pie charts, display bar charts side by side or stacked, overlay line graphs – and more!
- COMMS MODULE: Using a modem you can access services such as MicroLink and book rail or theatre tickets, send electronic mail, telex and telemessages in a flash – and more!

Mini Office II is a comprehensive integrated suite of programs setting a new standard in home and business software.

Most of the wide range of features – many of which are usually only available on software costing hundreds of pounds – are easily accessed by using cursor keys to move up and down a list of options and pressing Return to select.

Mini Office II takes over where the original, highly successful Mini Office left off, with numerous extra features, two additional modules, a program to convert existing Mini Office files to Mini Office II format, and a 60 page, easy to follow manual.

And it's at a price everyone can afford!

B, B+	Tape £14.95
	5.25" disc £16.95
Master	5.25" disc £19.95
Master Compact	3.5" disc £21.95

B, B+, Master, Master Compact	4 by 32k Rom board £59.95

INSTANT Mini Office II

Now available on a power-packed rom board!

- Four 32k roms on one board packed with 128k of super fast machine code.
- Easy to install – no soldering needed.
- Split second application selection.
- Frees valuable ram space for much more data.
- Fully compatible with the whole BBC Micro range, Aries or Watford shadow ram board, and tape, DFS, ADFS and ECONET filing systems.

The full selection of Mini Office II is available through BEEBUG Retail at MEMBER'S DISCOUNT. Please refer to your BEEBUG Retail Catalogue

DATABASE SOFTWARE

Europa House, 68 Chester Ro
Hazel Grove, Stockport SK7 5

PRINTWISE — professional looking documents with the minimum of fuss

The ideal companion for Wordwise, View and Inter-word

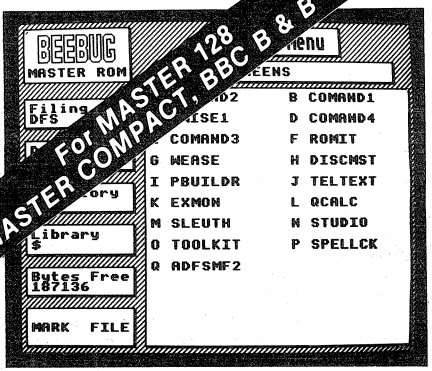
Printwise is a low cost publishing aid, and allows you to create professional looking magazines, leaflets, posters — the possibilities are endless. Also use it for 'near-letter quality' printout for correspondence. Simply take your text file, specify the font styles you require, and Printwise will do the rest.

No programming skills are necessary.

PRICE	MEMBERS PRICE
£30.00	£22.50

- 18 authentic fonts covering 4 standard typesizes.
- all fonts can be used in the same document, or even in the same line!
- italic, bold and condensed typestyles.
- proportional spacing, right justification, full indentation.
- powerful font designer.
- handles any length text files.
- suitable for all Epson compatible printers, and Shinwa CP80.

For MASTER 128 MASTER COMPACT, BBC B & B+



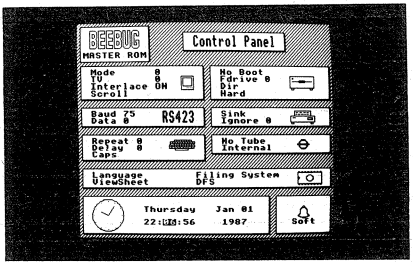
DISC MENU

A powerful disc menu giving complete control over disc files, and especially useful to ADFS users for easy movement through ADFS directories.

- Auto-run Basic and machine code programs.
- Automatically load Wordwise, View and Interword files.
- Scan ADFS directories, and step back to previous directory levels.
- Mark files for deletion, copying, re-naming etc.
- DFS and ADFS compatible.
- 40 or 80 column display.

DIARY & ALARM

A computerised diary for keeping track of appointments and important dates. Includes a monthly calendar display, allowing a diary to be kept for each day of the year. You can step backwards and forwards through the months and years, and set alarms for any dates.



CONTROL PANEL

A complete visual display of computer status allowing a whole range of settings to be examined and edited.

- Analogue clock display allowing clock to be adjusted.
- Comprehensive ROM/RAM display with options to insert, unplug and save ROMs.
- Alter configuration settings held in CMOS RAM (Master 128 only).
- Load/save status settings to disc.
- 40 or 80 column display.

SIDEWAYS RAM COMMANDS

A selection of commands to allow you to make more use of your sideways RAM. Use it as a printer buffer so that you can continue to use your computer for other work at the same time as printing long documents. It may also be used as a silicon disc, to load and save Basic or machine code programs to sideways RAM. A combination of both printer buffer and silicon disc can be selected.

DISC COMMANDS

The Master ROM contains a selection of disc commands to provide many useful features normally only available on the utility disc. Included are commands to backup ADFS discs, selectively copy files between the ADFS and DFS, catalogue whole ADFS discs, format, verify, and much more.

Although the Master ROM was designed for the Master 128, most features are available on the Master Compact, BBC B and B+. To make full use of the Master ROM, your BBC B or B+, should be fitted with the ADFS and sideways RAM.

BEEBUGSOFT

Master ROM

NEW

The Master ROM is a powerful new ROM developed to enhance the ADFS, sideways RAM and real time features of the Master 128. It has been further enhanced to enable users of the BBC Micro to benefit from its wide range of utilities.

Price £39.00

MEMBERS PRICE £29.25

Supplied on ROM with spiral bound manual.

Personal Ads

INTERWORD unused, latest version, with blank registration card £38; Watford 32K RAM board £40; APTL ROM board with RAM chips £30; Toolkit 1 £15, Spellcheck 1 £5. All boxed with manuals. DNFS ROM and booklet £10, Watford double BBC plinth £12. Upgrading to Master forces sale. Tel: Harlow 37527.

GRAPHICS ROM and manual £12. Stack L/Pen + Penpal disc £12. Watford Speech Synth + ROM £12. 20 various games £3 to £5 each, or will swap. Tel: (0704) 37821

ACORN Z80 second processor. All software as new, £200. Tel: (0727) 40674.

BEEBUG Magazine cassettes, November 1983 to October 1985 inclusive (20 off), £30. 22 blank cassettes £5. Assembly language programming book (Birnbaum) £5. Tel: Tonbridge (0732) 357757.

APTL sideways ROM board with booklet £15 plus p&p. BEEBUG back numbers Vol.1 to Vol.3 No 9, 25p each, 10 for £2. Tel: (0491) 872566, evenings.

KAGA colour monitor, medium resoluton, £100 ono. Tel: Brian Gard 01-274-8223 ext.23, daytime.

CUMANA double s/s switchable disc drive £45. Phillips monochrome (green) monitor £40. Various software: VIEW Version 3.0 (ROM + manual + strip + printer driver) £40, Disc Doctor £8, Wordwise £10, Inter Chart £10, Acornsoft Database (disc) £5, Beebug Design (80T disc) £5, Alligator Mk2 upgrade £8, Watford ROM Manager £5, Clares BROM £8, Watford NLQ ROM £5, New Generation Machine Code tutor (tapes) £5, Durell 'Combat Lynx' (tape) £4, Gemini Maillist (40T) £5, Gemini Beebcalc (40T) £5, Gemini Home Accounts (40T) £5, Acornsoft Money Management (tape) £4, Silversoft 'Index' database £8. All with manuals and inc. postage. Tel: 01-399-2865

BEEBUGSOFT Printwise £15. Exmon 1 £5. Watford Buffer & backup £5. CC Printmaster £8. All originals with manuals. Tel: (078571) 2104.

TEC DISC DRIVE, 40/80 double sided, no PSU £80. ARIES B32 shadow RAM board with B12 extension ROM board, all manuals and fittings cost £120+VAT, £85 no offers. BBC B Issue 7 Peartree 4 ROM board £250 ono. Tel: 01-831-2735, Mr.Garbett, office hours only.

TORCH Graduate Unit comprising of two 5.25" disc drive units & IBM compatible software £250 ono. Tel: Bradford (0234) 856050.

VIEW A2.1 ROM wih 'Into View' and 'View Guide' manuals, box included, £25. Will consider swapping for other hardware. Tel: Sheffield (0742) 899328 after 4.30 pm.

TURBO co-processor for Master £75. Various ROM, disc and cassette software plus a few books. All original and in good condition. Tel: Reigate (07372) 41033 after 5pm and weekends.

BBC B Watford DFS + 32K RAM board. APTL Board with 16K RAM, Solidisk RTC, Viewsheet, Spellcheck II, Wordwise Plus, Addcomm, Brom, Watford NLQ, £380. Will split ROMS. ARIES B20 £35. Computer village ROM board £30. Tel: 01-317-0797

WORDWISE Plus £25, Beebugsoft Toolkit ROM £10, both in original packaging with all manuals. Advanced Basic ROM Guide £5, Jeremy Ruston Compiler (cassette and book) and BBC Micro Revealed, £5 the pair. Silent Computers console with shelf for drives £20. Prism Modem 1000 (direct connect) £15. Collect or pay postage. Tel: Dunstable (0582) 601013.

BEEBUGSOFT Toolkit ROM, boxed + manual £12, Acornsoft Meteors and Snooker cassettes, boxed £4 each. Ultimate's Atic Atac cassette in original box £4. All as new, £20 the lot. Tel: Ottershaw (093287) 3435, evenings.

BBC B with DFS £150, ACORN Z80 processor full software and DNFS £175, APTL ROM board £15, Toolkit £10, Multiforth £20, CC Graphics ROM £10, Prism acoustic modem £5. Tel: Gloucester (0452) 21245 evesnings.

→34

Minerva Systems bring you another World First!

NEW

SYSTEM GAMMA

Imagine, you want a graph of your sales, but you also want a pie chart, histogram and a scatter graph on the screen simultaneously - AND you want to put text labels on the screen, AND everything in a specific place. Well, now it's a reality!

★ Completely definable - display multiple charts of varying types.

★ Place text or plot anywhere on the screen.

★ Statistics include standard deviation, line of best fit, etc.

Integrates fully with the System Delta range. Interfaces with BASIC.

AND, as you would expect, it is fully programmable for Users who require it.

SYSTEM DELTA

Still the most comprehensive & flexible Database System on the BBC.

The CARD INDEX program supplied with System Delta is simple to operate for everyone including a new User to the BBC, yet it's versatility and power are unrivalled. Office workers find it a pleasure to operate - just ask some of our business users - Rolls Royce, Range Rover, British Aerospace, ICI, CEGB, Smiths Instruments, Ciba-Geigy, British Gas, United Biscuits, British Telecom, British Museum or even the BBC themselves.

Reviewed as the simplest database to understand - now its even easier with our new issue manual.

Furthermore it is the only system on the BBC that links to a vast range of other programs including: Reporter, Mailshot, Inter/View Link, Invoicing, Sales, Purchase, Stock, Nominal, Video Rental, School Administrator, Hotelier, Newsagent.....

MINERVA TAKE THE RISC

Watch this space for Minerva's new RISC based product range which take Databases/Graphics into the 21st Century. - Step into the Future with the wise Goddess Minerva.

MINERVA SYSTEMS
69 Sidwell Street, Exeter, Devon EX4 6PH
Telephone: 0392 37756

Please supply Send details
System Delta with Card Index £64.95 System Gamma £45.95

Chq/Access No.:

Name :

Address :

MODE 7 SCREEN EDITOR

This keystrip is designed to be used with the Mode 7 Screen Editor on pages 23 to 28 of this issue. Either cut out the keystrip or photo-copy this page and place under the plastic strip on your micro when using this program.

Key f0	Key f1	Key f2	Key f3	Key f4	Key f5	Key f6	Key f7	Key f8	Key f9
BLACK BACKGROUND	NEW BACKGROUND	HOLD GRAPHICS	RELEASE GRAPHICS	NORMAL HEIGHT	DOUBLE HEIGHT	FLASH	STEADY	BLOCK CHARACTER	STAR.. COMMAND
RED GRAPHIC	GREEN GRAPHIC	YELLOW GRAPHIC	BLUE GRAPHIC	MAGENTA GRAPHIC	CYAN GRAPHIC	WHITE GRAPHIC	CONCEAL	CONTIGUOUS GRAPHICS	SEPARATE GRAPHICS
RED TEXT	GREEN TEXT	YELLOW TEXT	BLUE TEXT	MAGENTA TEXT	CYAN TEXT	WHITE TEXTR			

←32

Viewsheet £25, Wordwise Plus £25. Tel: Dr. McFetridge, Wokingham (0734) 775384.

SCRABBLE (40T) and Hobbit (40/80T) unwanted gifts £5 each + 50p p&p. Can sell separately. WANTED single 80 track drive for 'B' with DFS 0.9. Tel: Peter Sotheran, Cleveland (0642) 471662, Gold 87:YQQ093.

ROMS including AMX Pagemaker, AMX Max, Replay for the BBC Plus 1770, Icon Master & Toolkit Plus. Also a large amount of software on disc/tape, i.e. Music Island System, Citadel, Impossible Mission, £100 for everything. Contact: A. Cook, 54a Chichester Rd, Portsmouth, PO2 0AD.

ROMS for sale: Toolkit, Exmon II & Disc Doctor £12 each. Replay (8271 version) £15. Original documentation. Tel: M.May, 01-699-5087 after 6pm.

MASTER 128 including manuals. Printmaster and ViewStore ROMs, misc. software, CD800S, RX80F/T+, Zenith 122, all at £900. Z80 including software and manuals £250. Tel: Stoke-on-Trent (0782) 394411.

BBC 'B' with Acorn DFS, Aries shadow RAM and ROM expansion boards, twin 40/80 track disc drives (plinth style), lots of software such as ViewSheet, Exmon II, Spellcheck III etc, plus all back issues of Beebug and Micro User. As new condition £575. Tel: Peter Lindsley 01-998-6225, 6.30-9.30 pm.

HARRIER joystick £1, Advanced User Guide (brand new) £8, ADFS upgrade 1770 £15. Acorn Speech upgrade £20. Acornsoft Logo £20, Econet upgrade £20, Wordwise £15. All with full documentation and in original packing. Will haggle. Tel: John Bibby, Edinburgh (031) 664-3765.

PHILIPS LFH 290 Pocket Memo & LFH 505 Dictation transcriber, new, unwanted prize worth over £300, will accept £220 ono. Tel: Cambridge (0223) 357411.

MODEM for sale. CASE 400/22b (new in box). Specifications: full duplex, 2400 bps. or 1200 bps., BABT approved, complete with power supply and telephone cable (series 600). Further details available. Offers around £500. Tel: Keith, Luton 21396.

VARIOUS BBC games on cassette and disc from £1 to £10. Electron with tape recorder and software £100 ono. Tel: Lee, (0679) 21261 after 6 pm and weekends.

BBC B tapes: Drawing, Morse, Chemiplant, Jars £3. Programs 2, Frogger, Creative Graphics, Tree of Knowledge, White Knight MK12 £5 each. Tel: 01-500 7476

ACORN 1770 disc controller and DFS complete disc uprade for Model B £30,

WIN AN ARCHIMEDES

With this issue of BEEBUG you have an opportunity to win one of Acorn's new RISC machines, a brand new Archimedes 310 with colour monitor worth over £1100. This is the exciting prize that Acorn have generously made available. Full details of this competition are on page 57 of this issue and you have until 25th August to complete your entry. This is an opportunity not to be missed by all BBC micro users.

ARCHIMEDES

Classified Ads

"PUZZLE SOLVING FOR THE BBC COMPUTER". Solving puzzles with your computer is fascinating! The ideas in this little booklet will help you to do it. Cheques for £6.50 (including postage) to: P.G.Amey, 31 Hillmont Road, Hinchley Wood, Esher, Surrey KT10 9BA.

CHESS EXPERT SYSTEM. A complete storage, retrieval and analysis system for chess games with White Knight/Colossus interface. Master/Model B compatible. Available with 350K openings database. Send £1 (stamps accepted) for demonstration disc to Bernard Hill, Hawthorn Bank, Scott's Place, Selkirk TD7 4DP.

MABTech MASTER DEBUG ROM. Sophisticated Memory Editor and 65C12 Disassembler. Edit FS-RAM, MOS-RAM, Shadow RAM, ROM/RAM banks. Powerful memory search. £20. Send for details: MABTech, 2 The Grove, Ickenham, Uxbridge, Middlesex. Tel: (0895) 35365.

EVENTS

Acorn User Show,
Barbican Centre, London. 23-26th July 1987

PCW Show,
Olympia, London. 3-8th September 1987

Electron & BBC Micro User Show,
Old Horticultural Hall, Westminster. 13-15th November 1987

The Fourth High Technology in Education Exhibition,
Barbican Centre, London. 20-23rd January 1987

Note: We will, as usual, have a stand at the Acorn User Show in July.

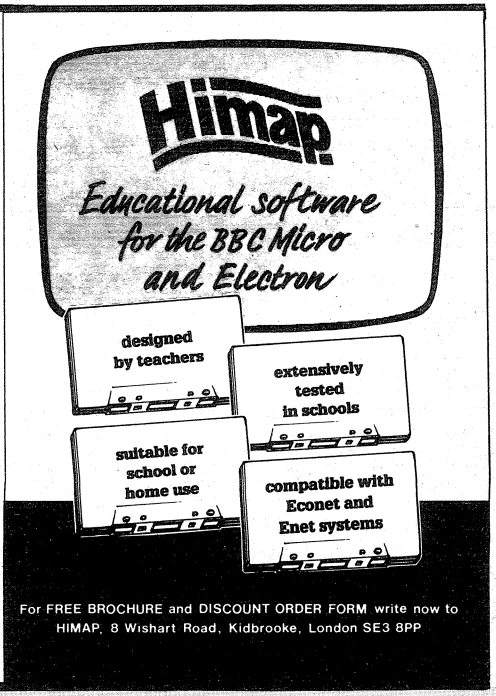
Advertising in Beebug

For advertising details, please contact

Yolanda Turuelo
on
(0727) 40303

or write to:

Dolphin Place
Holywell Hill
St. Albans AL1 1EX



BEEBUG MAGAZINE OFFERS

BEEBUG FILER — see this issue

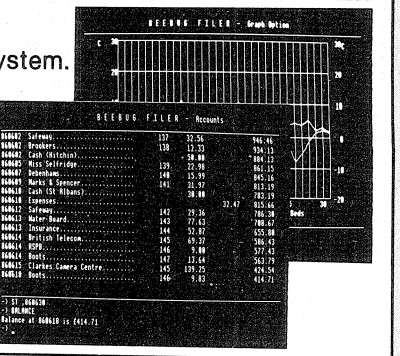
BEEBUG magazine's own database management system.

Suite of five programs including:

- ★ Top-level Menu Selector
- ★ Database Management including Mail-Merge
- ★ Graphics Option
- ★ Home Banking Option
- ★ Fast Sort Program

Supplied on dual 40/80 track disc — Instructions and program notes included.

Price £5.50 plus £1.00 post and packing.



BEEBUG Sideways RAM Module — see Vol.5 No.7

The cheapest and easiest way of adding sideways RAM to your Beeb

- ★ Plugs directly into any BBC ROM socket
- ★ Construction and software described in magazine

Available as a kit of parts for you to make up yourself, or ready made

Price £8.95 (kit) **£12.95** (ready made)

Post & packing £1.00 in each case

BEEBUG Master Alarm ROM — see this issue

- ★ Automatic clock and date alarm for the Master 128
- ★ Available in ROM for immediate use
- ★ As described in this month's magazine

Supplied on EPROM

Price £6.45 plus £1.00 post & packing

ORDER FORM

Name.....

Address .....

.....

.....

Membership Number.....

Signature .....

I enclose cheque for £_____

Please debit my Access/Visa No.

Please send:	Price
Filer Disc_____	£.....
BEEBUG Sideways RAM Kit____	£.....
BEEBUG Sideways RAM Module	£.....
BEEBUG Master Alarm ROM____	£.....
Post & Packing_____	£.....
Total_____	£.....

Send to: BEEBUG Mail Order, Dolphin Place, Holywell Hill, St. Albans, Herts AL1 1EX
Telephone Orders welcome. Tel: (0727) 40303

THE MASTER PAGES

Devoted to the Master Series Computers

This month's special Master Series features our Turbo ADFS menu. This relatively short machine code program works at lightning speed to select files, and generally move around within the ADFS directory structure. It uses a simple Mode 7 display with colour highlighted directories.

We have a review update this month on Master Cartridges, including the amazing MR8000 cartridge containing 64K of battery-backed RAM.

This month's Master Series Hints contains some useful ideas from members. But do please keep your contributions coming in.

37 Turbo ADFS Menu
40 Master Hints
41 New Master Cartridges

MASTER SERIES

Turbo ADFS Menu

Use this fast machine code ADFS menu by Alan Webster to select and run Basic programs, ROM images, or files from any language ROM — whether View, Wordwise, ViewSheet or Interword — and it works on the model B too.

In BEEBUG Vol.5 No.4 we published an ADFS menu for the Master by Peter Rochford. This well written piece of software has proved very popular amongst members. So why the need for a second menu? The menu described here will work on the Compact or the model B with ADFS as well as the Master 128; and it is written in machine code, which gives it a number of important advantages. Firstly, it operates much more quickly than our earlier menu, and secondly it is able to reside in the computer concurrently with other programs or data. This means that it may be called from Wordwise or View, and used to move around within the directory structure of the ADFS, and then return to the wordprocessor without the loss of your text (at least, within moderation - see later).

GETTING GOING

First of all, type in the program, and save it to disc before running it. After it has been run, you should save the assembled version into the library directory of each of your ADFS discs by copying the screen prompt. To use the menu, make sure that you have an ADFS disc in drive zero, then either type

```
CALL &6000
```

if the code is already resident in memory, or use

```
*TURBO
```

- assuming that you have saved the assembled code under the name "TURBO".

You will be presented with a display similar to that illustrated, with filenames in green and directory names in yellow. The cursor keys allow you to change directories. The "up" key takes you to the root directory, while "left" and "down" take you to the immediate parent directory or to the previously selected directory, respectively. To change drives, use keys "0" or "1". These two keys may also be used to change a disc in the currently selected drive.

FILE SELECTION

The primary purpose of the menu, of course, is the selection of files. This is achieved by pressing a key corresponding to the character adjacent to the required file. If the file is a Basic program, it will immediately be CHAINed. In all other cases, the software will follow a special convention which was described in the previous issue (page 43). If the menu detects that the chosen file is not a Basic program, it will store the filename of the selected file in memory at the bottom of the stack (at

&100), and attempt to EXEC in the first file which it comes across in the current directory with a name beginning with "!MENU".

It is up to the user to set up his own !MENU files to suit his specific requirements. The technique was described at some length in the previous issue, and sample files were given for Wordwise, View, Interword, Viewstore, Viewsheet and machine code. User customising can take any form, and allows you to incorporate function key definitions, date transference on a Master, printer configuration, etc, etc.

Unlike the Rochford and the Master ROM ADFS menus, which also follow this convention, the present menu does not specifically cater for the loading of ROM images. However, the !MENU file listed here easily overcomes this deficiency. If a ROM image is selected, this file will first execute *ROMS and then request the destination socket number (4-7 on a Master or Compact) before the *SRLOAD takes place.

ABOUT THE MENU

The menu takes up nearly 4 pages of RAM, and we have located it at &6000, below the mode 7 screen. If you are using shadow memory only, then it could be located at &7000 (alter line 1010), leaving a bit more space for resident programs and text. Of course, since it is located in normal user RAM it is possible to over-write it with a long program or text file. Conversely, calling the menu from Wordwise or View with a very long text file in the computer will result in the corruption of the file - so you need to take some care. If you expect to be using the menu in this way, it is probably best to experiment a little with a file that you don't mind losing in order to get the feel of what you can and cannot do.

The only way to be sure that corruption will never occur is to put the menu into sideways RAM/ROM - perhaps using the technique evolved by Bernard Hill (BEEBUG Vol. 5 Nos. 2, 3 & 4). If there is sufficient interest, we will cover this in a future issue.

!MENUroms File

```
OSCLI("ROMS"):P."RAM No?":A$=GET$:B$="*S
RLOAD "+$&100+" 8000 "+A$:OSCLI("KEY0"+B
$+"|M"):*FX138,0,128
```

```
  10 REM Program Turbo Menu
  20 REM Version B0.1K
  30 REM Author  Alan Webster
  40 REM BEEBUG  July 1987
  50 REM Program subject to copyright
  60 :
 100 ON ERROR GOTO 160
 110 MODE135
 120 PROCassemble
 130 PROCsave
 140 END
 150 :
 160 ON ERROR OFF:MODE 135
 170 REPORT:PRINT" at line ";ERL
 180 END
 190 :
1000 DEFPROCassemble
1010 code=&6000
1020 S%=&900:D%=&911:oscommand=&950
1030 FOR pass=0 TO 3 STEP 3
1040 P%=code
1050 [OPT pass
1060 .menu:LDA#4:LDX#1:JSR&FFF4
1070 LDA#22:JSR&FFEE:LDA#135:JSR&FFEE
1080 JMPadfs
1090 .drive:SEC:SBC#48:STA&7D
1100 LDY#255:.mnlp2:INY:LDAmount,Y
1110 STAoscommand,Y:CPY#6:BNEmnlp2
1120 LDA&7D:CLC:ADC#48:STAoscommand+5
1130 LDX#oscommand MOD256
1140 LDY#oscommand DIV256:JSR&FFF7
1150 JMPmainmenu
1160 .adfst:EQUS"ADFS":EQUB13
1170 .adfs:LDX#adfst MOD256
1180 LDY#adfst DIV256:JSR&FFF7
1190 :
1200 .mainmenu:JSRdisplay
1210 LDX#5:LDY#18:LDA#0:STA&7E:JSRgbpb
1220 LDA#1:STA&80:JSRfile
1230 LDA#31:JSR&FFEE:LDA#0:JSR&FFEE
1240 LDA#24:JSR&FFEE:LDY#255
1250 :
1260 .input:INY:LDAinptxt,Y:JSR&FFEE
1270 BNEinput:.inplp:LDA#127
1280 JSR&FFEE:.retent:LDA#21:JSR&FFF4
1290 JSR&FFE0:CMP#27:BEQ esc
1300 CMP#13:BEQretent
1310 CMP#48:BEQsetdrv:CMP#49:BEQsetdrv
1320 CMP#139:BEQup:CMP#136:BEQleft
1330 CMP#138:BEQdown:JMP testit
1340 .esc:LDA#126:JSR&FFF4:LDA#12
1350 JSR&FFEE:LDA#4:LDX#0:JSR&FFF4:RTS
1360 .up:LDX#upt MOD256:LDY#upt DIV256
1370 JSR&FFF7:JMPmainmenu:.upt
1380 EQUS"DIR $":EQUB13
1390 .left:LDX#leftt MOD256
1400 LDY#leftt DIV256:JSR&FFF7
1410 JMPmainmenu:.leftt:EQUS"DIR ^"
```

```
1420 EQUBl3:.down:LDX#downt MOD256
1430 LDY#downt DIV256:JSR&FFF7
1440 JMPmainmenu
1450 .downt:EQUS"BACK":EQUBl3
1460 .setdrv:JMPdrive
1470 .testit:JSR&FFEE:CMP#65:BCCnogood
1480 SEC:SBC#64:CMP&80:BCSnogood
1490 JMPgetfileinfo
1500 .nogood:LDA#7:JSR&FFEE:JMPinplp
1510 :
1520 .getfileinfo:STA&85:LDA#21
1530 JSR&FFEE:LDA#0:STA&80:STAS%+9
1540 STAS%+10:STAS%+11:STAS%+12
1550 .finfl:LDX#8:LDY#10:JSRgbpb
1560 DEC&85:LDA&85:BNEfinfl
1570 :                    \ Filename got!
1580 LDY#255:.fnamelp:INY:LDAD%+1,Y
1590 STAS%,Y:CMP#13:BNEfnamelp
1600 LDA#S%MOD256:STAD%:LDA#S%DIV256
1610 STAD%+1:LDA#0:STAD%+2:STAD%+3
1620 LDA#5:LDX#D%MOD256:LDY#D%DIV256
1630 JSR&FFDD:STA&80:LDAD%+7:CMP#&80
1640 BNEsrun:LDA#0:STA&8F:JMPtestdir
1650 .srun:LDA#1:STA&8F
1660 .testdir:LDA&80:CMP#2:BNEnotdir
1670 JMPnewdirectory
1680 .notdir
1690 LDA#4:LDX#0:JSR&FFF4
1700 LDA&8F:BEQchain
1710 LDA#6:JSR&FFEE:LDY#255
1720 .trlpl:INY:LDAS%,Y:STA&100,Y
1730 CMP#13:BNEtrlpl:LDX#exc MOD256
1740 LDY#exc DIV256:JSR&FFF7:RTS
1750 :
1760 .exc:EQUS"EXEC !MENU*":EQUBl3
1770 .chain:LDY#&D7:LDA#138:LDX#0
1780 JSR&FFF4:LDY#34:JSR&FFF4
1790 LDX#255:.chnlp:INX:STX&80
1800 LDAS%,X:CMP#13:BEQenchn:TAY:LDX#0
1810 LDA#138:JSR&FFF4:LDX&80:JMPchnlp
1820 .enchn:LDY#34:LDX#0:LDA#138
1830 JSR&FFF4:LDA#6:JSR&FFEE:LDA#138
1840 LDY#13:JSR&FFF4:RTS
1850 :
1860 .dirtxt:EQUS"DIR "
1870 .newdirectory:LDA#6:JSR&FFEE
1880 LDY#10:.dirlp:LDAS%,Y:STAD%+4,Y
1890 DEY:CPY#255:BNEdirlp:LDY#3
1900 .dirlp2:LDAdirtxt,Y:STAD%,Y:DEY
1910 CPY#255:BNEdirlp2
1920 LDY#14:LDA#13:STAD%,Y
1930 LDX#D% MOD256:LDY#D% DIV256
1940 JSR&FFF7:JMPmainmenu:RTS
1950 .inptxt:EQUS"Please enter your cho
ice :*":EQUB0
1960 .mount:EQUS"DIR :n":EQUBl3
1970 .gbpb:LDA#0:STAS%:LDA#D% MOD256
1980 STAS%+1:LDA#D% DIV256:STAS%+2
1990 LDA#0:STAS%+3:STAS%+4:STAS%+6
2000 STAS%+7:STAS%+8:LDA#1:STAS%+5
2010 STY&7F:TXA:LDX#S%MOD256
2020 LDY#S%DIV256:JSR&FFDl:LDY&7F
2030 INY:LDA#13:STAD%,Y:LDA&7E:CMP#0
2040 BEQfine:LDAD%+1:CMP#13:BEQfine
2050 LDX#(D%+1) MOD256
2060 LDY#(D%+1) DIV256:STXoscommand
2070 STYoscommand+1
2080 LDX#oscommand MOD256
2090 LDY#oscommand DIV256:LDA#5
2100 JSR&FFDD:CMP#2:BNEfine:LDA#127
2110 JSR&FFEE:LDA#131:JSR&FFEE
2120 .fine:LDY#0:.gbpbl:INY:LDAD%,Y
2130 CMP#13:BEQgbpb2:BMIgbpb3
2140 JSR&FFEE:JMPgbpbl:.gbpb2:RTS
2150 .gbpb3:PHA:LDA&7E:CMP#0:BEQgbpb4
2160 PLA:JSR&FFEE:JMPgbpbl:.gbpb4:PLA
2170 RTS
2180 :
2190 .display:LDA#12:JSR&FFEE
2200 LDY#255:.displ:INY:LDAbox,Y
2210 JSR&FFEE:BNEdispl:RTS
2220 .box:EQUBl50:EQUBl91
2230 ]:A$=STRING$(36,CHR$175)+CHR$239+C
HR$10+CHR$13+CHR$150+CHR$181+CHR$134+"Di
rectory :"+CHR$31+CHR$37+CHR$1
2240 A$=A$+CHR$150+CHR$234+CHR$10+CHR$1
3+CHR$150+CHR$253+STRING$(36,CHR$252)+CH
R$254+CHR$31+CHR$16+CHR$1
2250 $P%=A$:P%=P%+LENA$
2260 [OPT pass:BRK
2270 :
2280 .file
2290 LDA#0:STAS%+9:STAS%+10:STAS%+11
2300 STAS%+12
2310 .fileloop:LDA#13:STAD%+1:JSRcursor
2320 LDX#8:LDY#10:LDA#1:STA&7E
2330 JSRgbpb:LDA#135:JSR&FFEE
2340 LDAD%+1:CMP#13:BEQ outfile
2350 INC&80:JMPfileloop
2360 .outfile:LDA#127:JSR&FFEE
2370 JSR&FFEE:JSR&FFEE:RTS
2380 :
2390 .cursor:LDA#0:STA&82:LDA&80:STA&83
2400 .curlp:LDA&83:CMP#19:BCCcurl2
2410 SEC:SBC#18:STA&83:CLC:LDA&82
2420 ADC#13:STA&82
2430 .curl2:CLC:LDA&83:CMP#19:BCScurlp
2440 LDA#31:JSR&FFEE:LDA&82:JSR&FFEE
2450 LDA&83:CLC:ADC#3:JSR&FFEE:LDA&80
2460 CLC:ADC#&40:JSR&FFEE:LDA#130
2470 JSR&FFEE:RTS
2480 :
2490 ]:NEXT
2500 ENDPROC
2510 :
2520 DEFPROCsave
2530 A$=" *SAVE TURBO "+STR$~code+" "+S
TR$~P%+" "+STR$~menu
2540 PRINT''A$
2550 ENDPROC
```

MASTER SERIES

Master Hints

More hints for the Master and Compact rounded up by David Graham.

Automatic Date-Stamping (ADFS)

This hint automates the suggestion made by R.Sterry in the April Hints (BEEBUG Vol.5 No.10 p.57). The idea is to date-stamp a disc to indicate when it was last backed-up, by incorporating the date into the disc title. What I have done is to place all the instructions into an EXEC file called "DATE". When you execute the command *DATE, you are asked which drive you wish to date, then the date is automatically read from the Master's clock and entered onto the disc via the *TITLE command. The EXEC file may be created using *BUILD or using the Master Editor. It uses the COLOUR statements to blank out echoing of the file contents.

```
MO.4:CO.0
CO.7:INPUT'"Which drive to date: "d$:
OS.("MOU. "+d$):t$=MID$(TIME$,5,11):OS
.("TITLE "+t$)
```

Colin Pither

EXEC Files Updated

In EXEC Files Update (BEEBUG Vol.5 No.8 p.41) various ways of achieving user input in EXEC files were discussed. The most obvious was hinted at but not pursued. Entering a whole series of commands on a single line avoids the problem of the unstoppable EXEC file. You can even achieve multiple INPUT statements in the same file providing you keep to the same line, separating each statement with a colon, e.g.:

```
INPUT"Enter filename: "f$:INPUT"Which
directory: "d$:OS.("LOAD "+d$+"."+f$)
```

Remember, though, that this example performs a *LOAD, not a Basic LOAD.

Colin Pither

Clearing RAM Banks

The command *SRDATA, used for reserving sideways RAM, throws up the error message "RAM occupied" if the specified RAM bank contains a ROM image. There is no specific command to clear a ROM image, but you can create one quite easily with the following EXEC file (which you might call CLEAROM).

```
INPUT"ROM No.: "n%:!&900=0:OS.("SRWRIT
E 900 +1 8007 "+STR$n%):OS.("SRDATA "+
STR$n%)
```

This file, which should occupy a single line (no carriage returns until the end), may be created using *BUILD, or with the Master Editor. To clear a ROM, just type *CLEAROM, supply the ROM number, and the job is done. But beware, you cannot repeat the command on the same socket without issuing an intervening *SRROMS n.

Alastair Taylor

There is another way to achieve the same end. You can create a short blank file which can be loaded over the top of the header of the ROM image to clear it. To do this, type:

```
*SRSAVE blank 8000 +20 n
```

where n is the number of an empty RAM bank. Now, any time that you want to clear away a ROM image, type:

```
*SRLOAD blank 8000 n
*SRDATA n
```

David Graham

Editorial Concatenating

Concatenating Basic or assembler lines to reduce the length of a listing can be performed with ease using the Master Editor. Select f4 (Find String), and enter the following:

```
$*space^#/:return
```

Then hit "R" each time that you wish to concatenate, and "C" when you do not.

David Graham

MASTER SERIES

New Master Cartridges

David Graham reviews some of the latest Master cartridges on the market including Peartree's 64K battery backed RAM module.

The purchase of one or more ROM cartridges is an absolute must for most Master users. There is now an increasing variety of ROM software on the market for the 128; and while the machine does have three internal sockets into which ROMs may be plugged, using more than one of these results in the loss of available sideways RAM. Filling all three sockets (and adjusting the appropriate links) will, unbelievably, cost you 64K of RAM, which is half of the computer's available memory. ROM cartridges offer a cheap and simple alternative with no inherent RAM loss.

In BEEBUG Vol.5 No.9 we reviewed two cartridges from Peartree Computers, which have subsequently become very popular with Master users. There have been a number of interesting developments in the field since then, and the purpose of this review is to take a look at these.

CARE ELECTRONICS

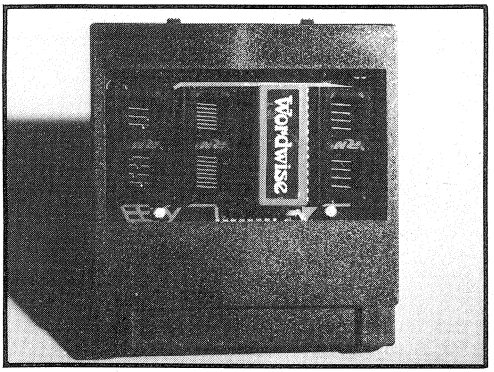
Care Electronics have recently entered the market with a range of four different types of cartridge for the Master. These are broadly similar to the Peartree units in shape, if not quite as elegantly finished. This is more than balanced however, by an important advantage. The Care units have been designed so that they will accept any of the new deeper ROM modules currently on the market. Computer Concept's Spellmaster, and BEEBUG's Master ROM are both supplied on a special carrier board with integral bank switching software to allow 32 and 64K ROMs to be used. These ROMs may not be used with the Peartree or Acorn cartridges, but slip neatly into the Care cartridges.

Two of the Care cartridges, the Dual and the Quad, have an essentially similar spec to Peartree's first two cartridges, the Pear 1 and the MR6000. Apart from accommodating the deeper ROMs, the Care Quad differs from the MR6000 in its use of switching. Both these devices house a pair of slide switches conveniently placed at the top of the cartridge. On the Peartree unit they are used for selecting alternate pairs of ROMs, for selecting a RAM mode (4 x 8K) and for optionally write-protecting the RAM. The Care unit may not be used with RAM, and its second slide switch is used to provide greater flexibility in selecting ROM banks.

MASTER CARTRIDGES AVAILABLE

CARE ELECTRONICS	
Dual Cartridge	**£8.97**
Single ZIF Cartridge	**£14.49**
Dual ZIF Cartridge	**£19.55**
Quad Cartridge	**£13.80**
PEARTREE COMPUTERS	
Pear 1 (Dual)	**£10.29**
MR6000 (Quad)	**£13.74**
MR7200 (32K RAM)	**£34.44**
MR8000 (64K RAM)	**£45.94**

Prices include VAT but not p & p



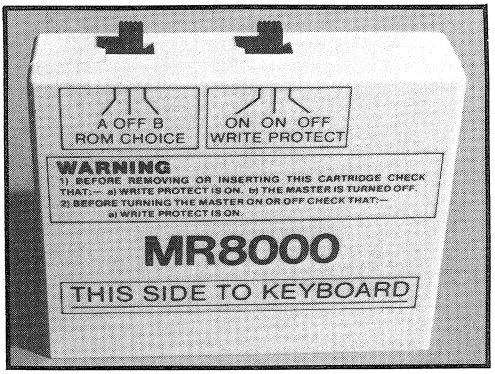
Two other Care cartridges are available: the "Single ZIF" and the "Dual ZIF". The latter contains two ZIF sockets, the former, one ZIF and one ordinary DIL socket. The zero insertion force sockets are of course very handy if you are intending to do a lot of ROM changing, though we do not have the space to review these units here.

PEARTREE COMPUTERS

Since our last review, Peartree have brought out two new cartridges for the Master. Both are sealed units containing banks of RAM. There is no battery back-up,

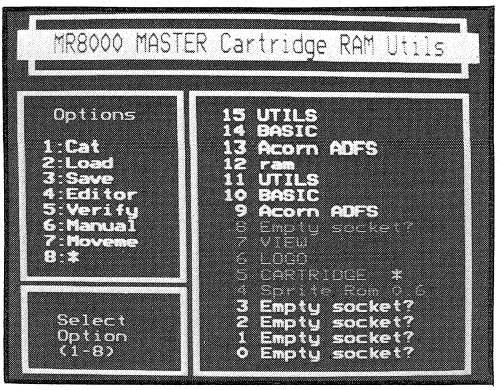
though the cartridge is supplied with a disc of RAM management utilities. The more expensive MR8000 contains 64K of battery-backed RAM. The remainder of this review is devoted to this revolutionary product.

The MR8000 is a neatly packaged unit broadly along the lines of the MR6000. Surprisingly, it comes with very few written instructions. All supporting software and documentation come on the RAM itself. Just plug in the cartridge, switch on your Master, and type *CART. This displays a well laid out menu with options for loading, saving and transferring ROM images, and for printing out the manual. The RAM is supplied as four 16K chips with a switch to select alternative pairs. There is a second switch which is used to write-protect the RAM, and this should be in write-protect mode whenever the computer is powered up or turned off. The battery back-up is provided by a lithium cell, giving an expected life of about 5 years. When it expires, the cartridge can be returned to Peartree for replacement.



In addition to the simple front panel software called with *CART, the ROM image supplied contains a number of extra star commands. The purpose of these is to implement a simple RAM filing system, allowing Basic programs to be saved and loaded to sideways RAM, and to catalogue or delete files. The addition of this software is a nice touch, and helps to open up the many possibilities provided by this cartridge.

The storing of Basic programs is perhaps the most obvious of applications for this unit. But if you want to have



access to a full 32K bank of RAM for this purpose, you will need to store the utility software somewhere else. The best solution would probably be to put it on EPROM, and keep it in the machine in a second cartridge. Another possible use for the cartridge is for the development of sideways ROM software. If you are either developing sideways software, or use sideways software which you add to from time to time, then the battery back-up facility provided here will be invaluable. No more tedious loading up of ROM images every time you switch on the computer.

Another use would be in diary, memo pad and alarm clock applications, where you really need battery-backed RAM to store alarm times, or diary messages actually IN the machine, rather than on disc - providing an excellent complement to the Master's battery-backed clock and calendar.

CONCLUSION

If you are in the market for a Master cartridge, then with so many sideways ROMs around, it makes sense to choose a four-socket switchable unit for most applications. If you need the extra ROM depth, then go for a Care Quad, otherwise try the smarter looking Peartree MR6000. But if you can afford it, and if you use sideways RAM to any extent, the MR8000 may well be a temptation too good to resist.

Peartree Computers
1 Blackstone Road,
Stukely Meadows
Huntingdon,
Cambs PE18 6EF.
Tel: (0480) 50595

Care Electronics
800 St Albans Road,
Garston, Watford,
Herts WD2 6NL
Tel: (0923) 672102

1st course

Plotting for Effect (Part 2)

This month, Mike Williams explains step-by-step how to use the PLOT command to colour fill a variety of different and irregular shapes.

In the First Course article last month we looked at how we could use the PLOT command, together with MOVE and DRAW, to produce various filled shapes. The technique involves the use of PLOT85, which draws a filled triangle and from this it is easy to produce filled rectangles and certain other filled shapes, such as circles as we saw last time. Master and Compact users have more PLOT commands available, which allow filled rectangles and circles to be drawn directly, rather than built up from triangles (PLOT101 and PLOT157 for example).

Now, although triangles, rectangles and circles have their place, it would be rather nice if Basic allowed us to fill in and colour any shape. In fact, BBC Basic does just that with some further variations of the PLOT command.

To fill an area of literally any shape on the screen is quite a complex task, and although this can be achieved just using Basic, we shall concentrate on some much simpler but nevertheless very useful fill routines. Master and Compact users are lucky to have access to a complete flood-fill routine (PLOT133 and PLOT141) which will also be described. Model B users requiring a comprehensive flood fill routine are referred back to Vol.5 No.3.

The basic fill instruction that we shall be using takes the form: PLOT77,x,y where the '77' specifies which PLOT option we are going to use, and 'x,y' refers to a point on the screen in the usual graphics co-ordinates. This instruction draws a line in the current drawing colour to the right and to the left of the point given. The line is continued in both directions until a point is reached which is not in the background colour, or the edge of the screen is reached.

Now this may well sound rather complicated, and you might also be wondering how this can be used to fill in an area, since the instruction just draws a straight line. Let's look at a very simple example:

```
100 MODE 2
120 GCOL 0,3
140 PLOT77,640,512
160 END
```

We shall be using mode 2 in all the examples this month as it allows us the choice of all 16 colours. Line 110 selects yellow as the drawing colour, and the background is black by default. The PLOT instruction in line 120 starts at the point (640,512), the centre of the graphics screen, and draws a yellow line horizontally right and left till it reaches the edges of the screen.

At this point it will be useful to introduce a procedure to draw an outline circle (in white), and of any radius and in any position on the screen.

```
1000 DEF PROCcircle(x,y,radius)
1010 VDU29,x;y;:GCOL0,7
1020 MOVE radius,0
1030 FOR angle=0 TO 2*PI STEP PI/8
1040 X=radius*COS(angle)
1050 Y=radius*SIN(angle)
1060 DRAW X,Y
1070 NEXT angle
1080 VDU29,0;0;
1090 ENDPROC
```

Master and compact users can simplify this considerably by replacing lines 1020-1070 with the two lines:

```
1020 MOVE0,0
1030  PLOT149,radius,0
```

Once you have typed this procedure into your micro, add this additional line to produce a complete program:

```
110  PROCcircle(640,512,200)
```

This very short program draws a white circle of radius 200 and with its centre at (640,512). The PLOT command in line 40 draws a yellow line as before, but you will find it now stops as soon as it

reaches the edge of the circle because this is no longer the background colour.

You might like to try some variations on this program, by changing the x and y co-ordinates given in the PLOT command. You should find that as long as you choose a point inside the circle, you get a horizontal yellow line inside the circle. If you choose a point that is outside the circle, then the line will reach at one end to the circumference of the circle, and at the other end to the edge of the screen (or indeed to both edges of the screen). The usefulness of this PLOT instruction lies in the fact that we do not need to know in advance where the ends of the line will be.

All we need to do now, to fill completely the inside of the circle, is to repeat the PLOT77 command from the bottom to the top (or top to bottom if you wish), by adding a FOR-NEXT loop with these two extra lines:

```
130 FOR Y%=312 TO 712
150 NEXT Y%
```

If you run this program you should find that the white circle is neatly filled with a yellow centre (of course!).

Now although this seems to work quite well, it is very easy indeed to improve the speed. To see how, we need to consider the question of screen resolution. Mode 2 that we have been using provides low resolution graphics, and the screen actually consists of 160 points horizontally by 256 points vertically, compared with the graphics co-ordinates that we use of 1280 by 1024. What this means is that in the vertical direction, for example, we have to move by a minimum of 4 graphics units in order to move 1 physical point on the screen. If we move in steps of less than 4 then we are still referring to the same physical screen point.

Thus in our mode 2 programs any vertical movement in steps of less than 4 is wasted, and in the horizontal direction we need steps of at least 8 to move physically on the screen. If we return to our previous program you should see that we can change line 130 to read:

```
130 FOR Y%=312 TO 712 STEP 4
```

If you run the revised program, you should find that you get exactly the same result as before, but four times faster!

Similar factors apply in all graphics modes. The minimum vertical step size is always 4 (physically 256 points on the screen), but the minimum horizontal step size varies according to the mode: 2 in mode 0, 4 in modes 1 and 4, and 8 in modes 2 and 5.

We have now seen enough to write a more general fill routine, which will fill not just a circle, but any shape, though with some limitations. We will write a procedure which will start from any point on the screen, and will fill the surrounding area. We will assume that the existing colour on the screen at the point specified is the background colour, and that the boundaries of the area to be filled will be marked by lines or areas of a different colour, or the edges of the screen, whichever are reached first.

In filling the circle we moved in one direction only, from bottom to top. With a more general approach, we no longer know where the boundaries are going to occur, and so we need to fill in the area in two sections, both starting from the point specified. The first section will be to fill in line by line moving vertically upwards from the starting point, and then to fill downwards from the starting point. In both cases we continue moving and filling until we encounter a point which is not in the background colour.

The processes of first filling up from a point, and then filling down from the point are essentially the same, so this has been written as a procedure called PROCfillupdown, which is then called twice in the main procedure, PROCfill. These two procedures are defined as follows:

```
1200 DEF PROCfill(colour,x,y)
1210 PROCfillupdown(colour,x,y,4)
1220 PROCfillupdown(colour,x,y-4,-4)
1230 ENDPROC
1240:
1300 DEF PROCfillupdown(colour,x,y,inc)
1310 newy=y:background=POINT(x,y)
1320 GCOL0,colour:GCOL0,128+background
1330 REPEAT
1340 PLOT77,x,newy
1350 newy=newy+inc
1360 UNTIL POINT(x,newy)<>background
1370 ENDPROC
```

Let's look first at the procedure PROCfillupdown, as this is the one which

does all the work. Notice the POINT function. This tells us the colour (as a number) of any point on the screen. The parameters x,y represent the starting point for filling an area, so the first task is to find the colour of this point and set it as the background colour using the GCOL command (lines 1310, 1320). Remember that background colours are always specified by adding 128 to the colour number. The other two parameters of this procedure are the colour to use for filling (colour) and the increment (inc) for filling up (4) or down (-4).

Since we no longer know in advance the start and end points we cannot readily use a FOR-NEXT loop this time, but this is ideally suited to a REPEAT-UNTIL loop. We shall repeatedly call the PLOT77 command, as we move up or down the screen, until the colour of the next starting point is different to the background colour, again detected by using the POINT function. Because we are using REPEAT-UNTIL we must ourselves program the incrementing of the new value of y (newy) to move up (or down) the screen. The fill procedure itself simply calls the other procedure twice, once to fill upwards from the starting point, and once to fill downwards from the starting point.

So now we have a general fill procedure that will enable us to fill in (or colour) any area, starting from any point on the screen. Some examples will soon expose the limitations of this procedure and you will begin to see why a really general fill routine is bound to be quite complex.

Our procedure will fill any circle (for example) provided the starting point is directly between the highest and lowest points of the the circle. Any other position, and the filling will stop as soon as the starting point for each line reaches the circumference vertically. You will also have to be careful using this fill procedure for concave areas, or you may well find 'holes' being left unfilled.

Despite these limitations, this procedure does have a lot of uses (a slightly extended version was used in last month's 'Night and Day' program), some of them quite amusing as we shall see in the trial example. This requires all three procedures that we have already been using, and has to be run to see the full effect.

The program first draws a large circle and colours it in. Two small circles are then added and coloured as eyes, and a third circle added and coloured as a nose. A REPEAT-UNTIL loop is then used to repeatedly open and close one eye to give the effect of winking. This is done by colouring in the eye with the same colour as the rest of the face, and then again with the colour of the eye. The starting points are carefully selected just inside the top and bottom edges in each case to give the realistic effect of an eye first closing and then opening again.

```
100 MODE 2
110 PROCcircle(640,512,400)
120 PROCfill(5,640,512)
130 PROCcircle(440,612,50)
140 PROCcircle(840,612,50)
150 PROCfill(4,440,612)
160 PROCfill(4,840,612)
170 PROCcircle(640,400,75)
180 PROCfill(1,640,400)
190 REPEAT
200 PROCfill(5,440,658)
210 PROCfill(4,440,568)
220 Z=INKEY(200)
230 UNTIL FALSE
240 END
```

On the Master and Compact, the PLOT133 comand can be used to replace the calls to PROCfill (and PROCfillupdown) used above. This is a comprehensive flood-fill command that does not have any of the limitations of our own fill routine.

Whichever system you have, I hope that you will be able to think of further ways to experiment with fill routines. Although comprehensive fill routines may seem attractive, they can be complex if you have to program them yourself. On the model B, the triangle plotting described last month will often be simpler to use, particularly when the shape of an area to be coloured is known in advanced. Otherwise, you may well find it easier to use a ready made routine, such as that from the previous issue of BEEBUG referred to before. Next month we will conclude this short graphics series by looking at more simple uses of the PLOT command. B

As well as the procedures and two demonstration programs listed here, the magazine cassette/disc for this month also contains a further example of the use of fill routines.

How to Write an Adventure Game

(Part 2) Jungle Adventure

Jonathan Temple finishes his description of the adventure game skeleton, and shows how to build it into a complete 'Jungle Adventure' for you to play.

This month we will complete the description of the adventure game skeleton. We will then examine the process of building a complete mini-adventure to demonstrate the use of the skeleton. You may, of course, just enjoy playing 'Jungle Adventure' as an adventure game in its own right. First of all, we need to consider the role of 'messages', the final component of our adventure game skeleton.

MESSAGES come after the nouns in the data lines. These should pose few problems - the only requirements are that they are stored in order from message 1 and are enclosed in quotation marks (") to avoid confusion. During the game, messages can be printed using PROCm(M) where M is the message number. Word-wrap is automatic for all messages.

Another useful feature is the set of five resident 'error' messages. These can be printed using PROCe(1-5); the actual messages can be seen in lines 3960-4000 of the skeleton program. All the data should be stored AFTER line 5000 and in the order: rooms, objects, verbs, nouns and messages. Otherwise, the actual line numbering for these data lines does not matter.

As well as rooms, objects etc., you can also include any number of general FLAGS in your game. These can set using PROCgf(F,V) where flag F is set to value V, and read using FNgf(F). There are also two associated procedures, PROCinc(F) and PROCdec(F), which will increment and decrement flag F respectively: these are useful for setting up counters. Note that the flag values can only be in the range 0-255. Incrementing past 255 produces 0; decrementing below 0 gives 255.

NEVER use new variables in your program, but instead use the flags feature. This avoids the risk of clashes with the skeleton program and, more importantly, the flags are saved using the SAVE/RESTORE feature, whereas any new variables will not.

Before writing any adventure games using the skeleton, you will need to know the "system variables" used by the program. These divide into two categories - ones which should be set and then left unaltered, and ones which are constantly altering throughout the game.

The first category includes all the variables in lines 120-130: r%, t%, f%, v%, n% and m%. These should be set by you to the number of rooms, things (objects), flags, verbs, nouns and messages used within your adventure. These variables MUST NOT be used or changed at any other point in the game.

There is another variable, TS in line 3100, which must be set at the start of the game and left unaltered. This number refers to the highest score possible, so if a game was scored out of 50 then TS would be set to 50.

Now come the variables which can be altered by you during the course of the game. First comes R, which holds the number of the player's current room. (If the player starts in a room other than 1, alter the value assigned to R in line 3100). The variable SC holds the player's score during the game. None of R, SC and TS should ever exceed 255. Lastly there is end%. This is set to zero at the start of each game: when you want to kill the player, set end% to TRUE. If the player manages to complete your game end% should be set to 2.

There are three more tasks involved in writing an adventure game using this skeleton, involving the setting up of three (presently empty) procedures PROCbefore, PROCleave and PROCafter.

First, PROCleave (lines 1560-1640). In your adventure you will probably want various things to happen when a player wishes to leave a certain room. These conditions should be tested for in PROCleave, using the variables R, D$ and F%. For instance, if a dragon blocks the south exit from room 2, you could use the condition IF R=2 AND D$="s" to test for the player going in this direction, so that the appropriate message is printed. If indeed the player is blocked from going the way he wants to (as in this example) and so must stay in the present room, the variable F% should be set to zero.

PROCbefore (1520-1540) and PROCafter (2500-2550) are two procedures which are called before and after a command is typed, and their use will be described later. We now come to the implementation of 'Jungle Adventure'.

JUNGLE ADVENTURE

The date is 1953: the place, a small base camp near the source of the Zambesi river. You are on a dangerous expedition through the heart of the jungle, with your only companion the experienced explorer Colonel Gordon Smithers.

After setting down camp for the day Smithers decides, after a long argument with you, to make a quick foray into the jungle surrounding your new camp. However, by nightfall Smithers has not returned. You wait anxiously for the morning, and he is still missing. The only thing to do now, you decide, is go into the jungle after him. Needless to say, your task in 'Jungle Adventure' is to find your companion and so complete the game.

The adventure accepts most of the usual adventure commands - TAKE, DROP, EXAMINE, WEAR etc. plus a few others besides. The commands should be entered in the usual verb/noun format, for instance GET JAR or EXAMINE PIPE. The state of the game at any time can be saved to tape or disc using the commands SAVE <filename> and LOAD <filename>.

ENTERING THE PROGRAM

To play the game you will need to have last month's adventure skeleton - by adding the lines presented this month you will complete the game 'Jungle Adventure'. The full game appears on this month's cassete/disc. When entering the program you MUST keep strictly to the line numbering - otherwise you will run into all sorts of problems!

IMPLEMENTATION

Firstly, those wishing to play the game seriously should not read the following section until they have, as otherwise it may spoil the game for them! The game was really written to demonstrate to people wishing to write their own adventures using the skeleton, how it is done.

The first thing to do when writing your own adventure, however you intend to program it, is to draw it all out on paper. That means drawing a map and outlining a plot, and then listing in detail the rooms, objects, verbs, nouns and messages. Most games also require the use of flags, for instance to signal whether a door is open or closed. These flags should also be worked out on paper.

Once I had done this, I started thinking about how actually to program the game. The first thing to do is set the system variables in lines 120-130, as explained earlier.

The data for the rooms, objects, verbs, nouns and messages was then entered into DATA lines, again as described previously. Then came PROCbefore. This procedure handles the things in the adventure which may happen before the player actually types a command in, and I had only one line to add to this: if the player was in the final room 12, he must have completed the game - so set his score to 50 and set end% to 2, so that the game ends.

Next I set up PROCafter, which performs a similar function to PROCbefore but for things which may occur after the player's command has been interpreted and acted on. PROCafter I used for handling the player's death from a snake bite (further details are given below).

The next - and most difficult - part was to write a short routine for each verb included in the adventure. For instance, there had to be a routine for getting objects (GET/TAKE) and another for dropping them (DROP/THROW).

Once these routines had been written I entered their line numbers into the the ON...GOSUB statement at line 1400. The

table below shows the verbs, and the line numbers for the corresponding routines. All that was then required was the final testing and debugging of the game.

Verb No.	Line(s)	Routine
1	1680-1740	GET/TAKE
2	1760-1800	DROP/THROW
3	1820-1860	FILL
4	1880-1930	EMPTY
5	1950-2000	BLOW
6	2020-2060	DIG
7	2080-2140	DRINK
8	2160	EAT
9	2180-2220	EXAMINE
10	2240-2280	WEAR
11	2300-2360	KILL
12	2380-2400	SWIM
13	2420-2440	CROSS
14	2460-2480	HELP

By going carefully through the different verb routines in this month's program, and referring back to last month's article, you should be able to work out how to write your own verb routines. If you are not entirely sure on some points, just experiment until you get the hang of it.

It may now help to look at how various aspects of the game have been implemented. For instance, if the player tries to GET the jar whilst in the same room as the native, the player is quickly killed. Bearing in mind that noun 2 refers to the jar, and object 6 is a native who can have one of two messages, 8 or 9 (message 8 is "an angry native" whilst message 9 is "a dead native"), you should be able to see how this has been programmed in line 1730.

The native can be killed by blowing down the cane - actually a blowpipe armed with one deadly dart! Study the BLOW routine (and especially line 1980) and you should see how this has been programmed (noun 1 is the cane and room 5 the native's hut). The routine for blowing the pipe also uses the cane's object flag, which is set to zero until the pipe is blown and the dart fired. This ensures that the dart can be fired only once.

A second problem in the game is the snake. You are bitten by this evil-looking creature should you try to enter Smithers tent, which is south of room 10. For this we can use PROCleave, testing for entering the tent, using something along the lines of IF R=9 AND D$="s".

Once you have been bitten by the snake, you have only a few moves in which to drink the antidote. General flag 1 is set to 1 after the snake bite (PROCleave, line 1630) and then used as a counter of the number of moves since you were bitten. Should the counter reach 4 the player dies; a warning message is printed just beforehand, when the counter reaches 3. This is all handled by PROCafter (lines 2510-2540). Should the player drink the antidote (the potion from the native's hut), then flag 1 is reset back to zero (look at line 2130, in the DRINK routine) and the player is safe.

LAST WORD

It will take some perseverance, and careful study of 'Jungle Adventure' and these two articles, before you become proficient in writing adventures using the skeleton. It is well worth it, however, as once you have mastered the skeleton you will be able to quickly and easily transfer your plots into a complete game.

Although a finished game would be very reasonable for publication in a magazine, it could never really be large enough to be up to commercial standards. This is because the skeleton presented here (to keep it short, and suitable for magazine games) stores everything twice, once in a DATA statement and then in a Basic array, rather than working 'from memory' as a commercial adventure would do.

For someone who wants to go further, a suitably more advanced skeleton for producing commercial-quality games (along with an interesting chapter on creating good plots) can be found in Peter Killworth's definitive book on the subject - "How to Write Adventure Games" at £5.95 published by Penguin/Acorn.

```
 10 REM Program Jungle Adventure
 20 REM Version B1.0
 30 REM Author  Johnathan Temple
 40 REM Beebug  July 1987
 50 REM Program subject to copyright
 60 :
120 r%=12:t%=7:f%=1
130 v%=18:n%=18:m%=36
```

```
 1400 ON V GOSUB 1680,1760,1820,1880,195
0,2020,2080,2160,2180,2240,2300,2380,242
0,2460
 1530 IF R=12 SC=50:end%=2
 1570 IF R=7 IF D$="w" IF FNom(7)=14 PRO
Cm(30):F%=0
 1580 IF R<>4 OR D$<>"n" THEN 1610
 1590 PROCm(34):IF FNor(4)=0 AND FNof(4)
 PROCm(35) ELSE PROCm(36):end%=TRUE
 1600 ENDPROC
 1610 IF R<>10 OR D$<>"s" ENDPROC
 1620 IF FNgf(1) PROCm(27):end%=TRUE:END
PROC
 1630 PROCm(31):PROCgf(1,1)
 1680 IF N=7 IF FNom(2)=2 N=2
 1690 IF N=8 IF FNom(2)=3 N=2
 1700 IF N=8 IF R=4 PROCm(11):RETURN
 1710 IF FNhere=0 RETURN
 1720 IF N>4 PROCe(1):RETURN
 1730 IF N=2 IF FNor(6)=R IF FNom(6)=8 P
ROCm(10):end%=TRUE:RETURN
 1740 PROCget(N):RETURN
 1750 :
 1760 IF N=7 IF FNom(2)=2 N=2:GOTO1880
 1770 IF N=8 IF FNom(2)=3 N=2:GOTO1880
 1780 IF FNheld=0 RETURN
 1790 IF N=4 PROCof(4,0)
 1800 PROCdrop(N):RETURN
 1810 :
 1820 IF N<>2 PROCe(1):RETURN
 1830 IF FNheld=0 RETURN
 1840 IF FNom(2)<>4 PROCe(4):RETURN
 1850 IF R<>4 PROCe(5):RETURN
 1860 PROCom(2,3):PRINT"OK":RETURN
 1870 :
 1880 IF N<>2 PROCe(1):RETURN
 1890 IF FNheld=0 RETURN
 1900 IF FNom(2)=4 PROCe(4):RETURN
 1910 IF FNor(7)=R IF FNom(7)=14 IF FNom
(2)=3 PROCm(13):PROCom(7,15):SC=SC+10:GO
TO1930
 1920 PROCm(12)
 1930 PROCom(2,4):RETURN
 1940 :
 1950 IF N<>1 PROCe(1):RETURN
 1960 IF FNheld=0 RETURN
 1970 IF FNof(N) PROCm(16):RETURN
 1980 IF R=5 IF FNom(6)=8 PROCm(18):PROC
om(6,9):SC=SC+10:GOTO2000
 1990 PROCm(17)
 2000 PROCof(1,1):RETURN
 2010 :
 2020 IF E<>1 IF N<>9 PROCe(1):RETURN
 2030 IF E=2 IF R<>6 PROCe(2):RETURN
 2040 IF R<>6 PROCm(19):PROCor(3,255):RE
TURN
 2050 IF FNor(4)=255 PROCm(20):PROCor(4,
6):SC=SC+10:RETURN
 2060 PRINT"OK":RETURN
 2070 :
 2080 IF N<>7 IF N<>8 PROCe(1):RETURN
 2090 IF N=8 IF R=4 PROCm(21):RETURN
 2100 IF (N=7 AND FNom(2)<>2) OR (N=8 AN
D FNom(2)<>3) PROCe(2):RETURN
 2110 IF FNor(2)<>0 PROCe(3):RETURN
 2120 PROCm(21):PROCom(2,4)
 2130 IF N=7 IF FNgf(1) PROCm(22):PROCgf
(1,0):SC=SC+10
 2140 RETURN
 2150 :
 2160 PROCe(1):RETURN
 2170 :
 2180 IF N>4 PROCe(1):RETURN
 2190 IF FNavail=0 RETURN
 2200 IFN=1 PROCm(24):RETURN
 2210 IFN=3 PROCm(25):RETURN
 2220 PROCm(23):RETURN
 2230 :
 2240 IF N<>4 PROCe(1):RETURN
 2250 IF FNheld=0 RETURN
 2260 IF FNof(4) PROCe(4):RETURN
 2270 PROCof(4,1):IFSC<45 SC=SC+5
 2280 PRINT"OK":RETURN
 2290 :
 2300 IF N=10 IFR<>4 PROCe(2):RETURN
 2310 IF N=10 GOTO2440
 2320 IF N<>5 IF N<>6 PROCe(1):RETURN
 2330 IF FNhere=0 RETURN
 2340 IF FNom(N)=9 PROCe(4):RETURN
 2350 IF N=5 PROCm(27) ELSE PROCm(10)
 2360 end%=TRUE:RETURN
 2370 :
 2380 IF E=2 IF N<>11 PROCe(1):RETURN
 2390 IF R<>4 PROCe(5):RETURN
 2400 GOTO2440
 2410 :
 2420 IF N<>11 PROCe(1):RETURN
 2430 IF R<>4 PROCe(2):RETURN
 2440 PROCm(26):end%=TRUE:RETURN
 2450 :
 2460 IF R=4 PROCm(28):RETURN
 2470 IF FNor(1)=0 PROCm(24):RETURN
 2480 PROCm(29):RETURN
 2490 :
 2500 DEFPROCafter
 2510 IF FNgf(1)=0 ENDPROC
 2520 PROCinc(1):IF FNgf(1)=3 PROCm(32)
 2530 IF FNgf(1)=4 PROCm(33):end%=TRUE
 2544 ENDPROC
 2550 :
 3100 R=1:SC=0:TS=50
 5010 DATA "in a small clearing, deep in
 the jungle.",0,0,2,0,0,0
 5020 DATA "in the jungle.",0,8,2,1,0,0
 5030 DATA "in the jungle.",2,0,4,9,0,0
 5040 DATA "on the bank of a fast-flowin
g river filled with evil-looking alligat
ors. To the north is a cliff, into which
 a narrow, unsafe-looking tunnel leads."
,12,0,0,3,0,0
```

```
5050 DATA "inside a small mud hut, belo
nging to a native.",0,9,0,0,0,0
5060 DATA "inside the cave, which has a
soft sandy floor.",0,0,7,0,0,0
5070 DATA "outside a cave, nearly hidde
n by the undergrowth.",0,0,8,6,0,0
5080 DATA "in the jungle.",8,0,3,7,0,0
5090 DATA "in the jungle.",5,10,0,2,0,0
5100 DATA "outside a tent, presumably a
bondoned here by Smithers.",9,11,0,0,0,0
5110 DATA "inside a narrow, cramped ten
t.",10,0,0,0,0,0
5120 DATA "in a narrow niche at the end
of the tunnel. Smithers lies here, inju
red. You have found him!",0,0,0,0,0,0
600 DATA 8,1,0,5,2,0,11,5,0
610 DATA 255,6,0,10,7,0,5,8,0,7,14,0
7000 DATA get,1,2,take,1,2,drop,2,2,thr
ow,2,2,fill,3,2,empty,4,2,blow,5,2,dig,6
,0,drink,7,2,eat,8,2,examine,9,2
7010 DATA wear,10,2,kill,11,2,hit,11,2,
attack,11,2,swim,12,0,cross,13,2,help,14
,1
8000 DATA cane,1,pipe,1,blowpipe,1,jar,
2,spade,3,helmet,4,pith,4,snake,5,native
,6,potion,7,water,8,sand,9,alligator,10,
river,11,fire,-2,hut,-1,tent,-1,dart,-1
8010 DATA "a long hollow cane"
8020 DATA "a jar containing a sickly-sm
elling potion"
8030 DATA "a jar full of water"
8040 DATA "an empty jar"
8050 DATA "a small rusty spade"
8060 DATA "a pith helmet"
8070 DATA "an evil-looking snake"
8080 DATA "an angry native"
8090 DATA "a dead native"
8100 DATA "The native jumps up and kill
s you!"
8110 DATA "The water just trickles thro
ugh your hands."
8120 DATA "The contents spill onto the
ground and evaporate."
8130 DATA "The water falls onto the fir
e, which quickly dies down."
8140 DATA "a blazing fire"
8150 DATA "the smouldering remains of a
fire"
8160 DATA "Nothing seems to happen"
8170 DATA "A small dart shoots out into
the undergrowth!"
8180 DATA "A dart shoots out and hits t
he native...the curare poison quickly ta
kes effect, and he dies."
8190 DATA "The rusty spade shatters on
the hard ground."
8200 DATA "You discover a pith helmet,
hidden here by Smithers!"
8210 DATA "It tastes horrible...!"
8220 DATA "However, it seems to have ac
ted as an antidote against the snake bit
e."
8230 DATA "You see nothing special"
8240 DATA "The cane appears to be a blo
wpipe of some sort."
8250 DATA "The spade is so rusty it wil
l break if used on hard ground!"
8260 DATA "The alligators chew you to p
ieces!"
8270 DATA "The snake bites you again an
d you quickly die!"
8280 DATA "You don't have to cross the
river."
8290 DATA "You're on your own here!"
8300 DATA "The fierce flames beat you b
ack!"
8310 DATA "The snake suddenly lunges fo
rward and bites you!"
8320 DATA "The snake's venom is slowly
taking effect..."
8330 DATA "You slowly die, poisoned by
the snake's venom."
8340 DATA "As you move north, there is
a sudden rumble above you, and a rock fa
lls..."
8350 DATA "only to be deflected by your
pith helmet."
8360 DATA "hitting your head and killin
g you instantly!"
```

← Quadric Surfaces

```
10200 DATA cone,SQR(X*X+Y*Y)
10210 DATA -12,12,.8,-12,12,1.5
10220 DATA 16,20
10230 :
10300 DATA hyperbolic paraboloid,Y*Y*.8-
X*X
10310 DATA -5,5,.4,-5,5,.4
10320 DATA 48,18
10330 :
10400 DATA hyperboloid of 1 sheet,SQR(X*
X+Y*Y-8)
10410 DATA -8,8,.25,-8,8,1
10420 DATA 16,30
10430 :
10500 DATA hyperboloid of 2 sheets,SQR(Y
*Y-X*X-1)
10510 DATA -20,20,.4,-20,20,2
10520 DATA 10,16
10530 :
10600 DATA not a quadric surface,X*X*X-3
*X*Y*Y
10610 DATA -3,3,.1,-3,3,.15
10620 DATA 80,6
10630 :
10700 DATA ?,EXPRESSION IN X AND Y
10710 DATA xmin,xmax,xstp,ymin,ymax,ystp
10720 DATA hscale,vscale
```

EXPLORING ASSEMBLER

(Part 2)

A series for complete beginners to machine code by Lee Calcraft

This month: The 6502 Register Set
Load and store operations
Subroutines and branching

Last month we introduced this series by taking a broad look at the principles behind assembly language programming. This month I want to introduce a small part of the 6502 instruction set, and to put these instructions to work in one or two simple examples. But before dealing with the instructions themselves, a brief discussion of the 6502's complement of registers will help to set the context.

THE REGISTER SET

Compared to the Z80, and to recent 16 and 32 bit processors, the 6502 has a very limited set of internal registers. There are in fact just six. See Table 1. The accumulator (sometimes referred to as the "A register") and the X and Y registers are the only ones which may be directly and freely manipulated by the user. Of these, the accumulator, with its large related command set, is by far the most important. Thus, while there are instructions to increase, decrease and compare values contained in the A, X and Y registers, only the accumulator can perform addition, subtraction, and logical operations. Because of this, the accumulator is used for most read, write, logic and arithmetical functions, while the X and Y registers tend mostly to be used as loop counters, memory offsets and so on.

Accumulator	8 bit
X Register	8 bit
Y Register	8 bit
Status Register	8 bit
Stack Pointer	8 bit
Program Counter	16 bit

Table 1
6502 internal registers

Of the remaining three registers, the Status Register is the one with which the programmer tends to have most direct contact. This holds seven "flags" which allow the programmer to test the result of various operations. A FLAG incidentally is just an indicator which usually takes the value of ONE or ZERO depending on whether a given condition has or has not occurred.

The fifth register, called the Stack Pointer, serves to keep track of an area of memory called the Stack which the 6502 processor, uses as a temporary storage area. Lastly we come to the Program Counter. This register holds the address of the next instruction to be executed. Except when using fairly sophisticated techniques, these last two registers are invisible to the user.

BITS AND BYTES

You will note from Table 1 that the first 5 registers are "8 bit" registers, while the last is "16 bit". A BIT stands for BInary digiT. This is the fundamental unit in all current digital computers. A bit can either be ON or OFF, and this is represented by a ONE or a ZERO respectively. We do not have the space here to digress too far into the fundamentals of binary arithmetic; neither is it necessary at this point. Table 2 gives a number of decimal, hex and binary equivalents. And it is really only necessary, for the time being, to appreciate that the highest number which can be represented by eight binary bits is 11111111. This is equivalent to 255 decimal or &FF hex.

decimal	hex	binary
1	1	00000001
2	2	00000010
3	3	00000011
4	4	00000100
15	F	00001111
16	10	00010000
127	7F	01111111
128	80	10000000
255	FF	11111111

Table 2
Some binary numbers
with decimal and hex equivalents

All memory locations in the BBC micro, just like the 6502 registers themselves, are eight bits wide, so these too can hold integers of up to 255 decimal. Eight bits

make up one BYTE, so the memory locations in the Beeb are often referred to as bytes. We said that just one of the 6502's registers, the Program Counter, was 16 bits wide. This is necessary to allow the 6502 to "address" or specify an adequate number of memory locations. In fact it can address any location from:

00000000 00000000 to 11111111 11111111
or 0 to 65535 decimal
or 0 to &FFFF hex

The total number of addressable locations, or bytes, is thus 65536 decimal, or 64K (where 1K=1024 decimal).

It is for this reason that the 6502 can only make direct use of a maximum of 64K (64 kilobytes) of memory. On the BBC computer this is split equally between RAM and ROM. The Master series use tricks to bring into play 128K of RAM, but this cannot all be directly addressed, with consequent limitations on the way in which it can be used. Just for the sake of comparison, Acorn's new RISC chip can address up to 64 megabytes of RAM! With that sobering thought it is time to move on to the 6502 instruction set.

LOAD AND STORE

One of the most fundamental pairs of instructions in any microprocessor instruction set involve the reading and writing of registers and memory locations. The 6502 has a pair of instructions for each of its three user registers.

LDA LoaD Accumulator
STA STore contents of Accumulator
LDX LoaD X register
STX STore contents of X register
LDY LoaD Y register
STY STore contents of Y register

In each case the register may be loaded either with a number (any integer between 0 and 255), or with the contents of a memory location. In the case of the store operation, the destination is always a memory location.

Let us look at some examples.

1. LDA #127:STA 6400
2. LDA #&F0:STA &1900
3. LDX &70 :STX &71

The first pair of instructions, separated by a colon, loads the accumulator with the decimal number 127. Remember from last month that it is the hash sign (#) which determines that it is the NUMBER 127, and not the CONTENTS of location 127 which is to be loaded. The accumulator is loaded with the number 127, and this number is stored in memory location 6400 decimal. The net result of this pair of instructions is that both the accumulator and memory location 6400 decimal will contain the number 127 decimal. The second pair of instructions has exactly the same effect, but both the value to be loaded, and the address at which the number is to be stored are expressed in hexadecimal.

The third pair of instructions work with the X register rather than the accumulator. Note that the first of this pair loads &70 (not #&70). This means that the X register is loaded with the CONTENTS of memory location &70 (which may be any integer between 0 and 255), rather than with the actual number &70 (112 decimal). The second instruction of the pair stores the number in the adjacent memory location, &71. In other words, the effect of this pair of instructions is to duplicate the contents of memory location &70 in &71.

Listing 1

```
100 REM LOAD & STORE OPERATIONS
110 REM Filename Assem22
120 oswrch=&FFEE
130 FOR pass=0 TO 1
140 P%=&900
150 [
160 OPT pass*3
170 LDA #&41:STA &70
180 LDX #&58:STX &71
190 LDY #&59:STY &72
200 LDA &70:JSR oswrch
210 LDA &71:JSR oswrch
220 LDA &72:JSR oswrch
230 RTS
300 ]
310 NEXT
320 CALL &900
```

Take a look at listing 1. This assembler listing is similar in many respects to the one which we used last month. But it has been expanded to include the instructions which we have been looking at here. First of all, try to work out what it will do when executed. As a clue, remember that &41 is the ASCII value of the letter "A", and &58 and &59 are ASCII "X" and "Y" respectively in hex. Three calls are made to the operating system entry point OSWRCH. This, remember, prints the contents of the accumulator as an ASCII character.

To check your theories, you could type in the listing, and save it to tape or disc. When you run the program, it will assemble, and then automatically execute (because of the CALL statement in line 320). If all is well, it will print the letters AXY on the screen, then return you to Basic.

The first pair of instructions load the value &41 into memory location &70. The second pair (using the X register this time) load the value &58 into location &71, and the third (using the Y register) load &59 into &72. The accumulator is then loaded with the contents of the first of these locations (redundantly, as it happens, since it already contains this value), and a subroutine jump is made to OSWRCH. This prints "A", since the contents of &70 is ASCII "A". The next pair of instructions print "X", since &71 contains ASCII "X". Lastly the "Y" is printed.

SUBROUTINES

To illustrate the writing of subroutines, we could include a simple subroutine in this piece of code to print a space between the letters. We could call this subroutine "space". To do this, add the following lines to Listing 1:

```
205 JSR space
215 JSR space
240 .space
250 LDA #&20
260 JSR oswrch
270 RTS
```

The first two lines here just execute a Jump to the SubRoutine labelled "space". The last four lines define the routine itself. The first of the four lines contains the LABEL "space".

Labels are used for a number of purposes in assembler code. They always take the form of a dot followed by a string of characters. Labels may be placed on a line by themselves, or be followed by other assembler instructions. In this case they must be immediately followed by either a space or a colon. In many respects they are like Basic variables, but they take the value of the address at which they fall in the assembly. To test this, RUN the listing, then type:

```
PRINT space
```

You should get the answer 2338 decimal (&92 hex). This is the decimal value of the address of the start of the subroutine "space". The effect of the remaining three lines is to load the accumulator with the value &20 (32 dec), which is the ASCII code for a space, then a JSR is made to OSWRCH. The last instruction is a ReTurn from Subroutine.

CONDITIONAL BRANCHES

So far we have considered linear program flow, where a program proceeds linearly from one instruction to the next. Subroutines constitute a temporary branch from the linear sequence. But there are two other types of branching possible. The simplest is an unconditional jump. It has a similar effect to the Basic GOTO statement, except that the destination supplied by the user must be a memory location (or a label signifying one) rather than a line number. The mnemonic is JMP, and the operand (i.e. its accompanying parameter) must always be an address in memory, either given as a label, a Basic variable or a straight address. For example, you could make the program considered earlier repeat forever by adding:

```
225 JMP start
```

and labelling the start of the program with:

```
165 .start
```

The conditional branch is a far more frequently used device, for reasons which will soon become obvious, though it is much more limited in the range of memory which it can address (see next issue). The 6502 has eight conditional branch instructions, and we will kick off with just one pair:

```
BEQ   Branch if EQual to zero
```

and

```
BNE   Branch if Not Equal to zero
```

At first sight you might think that these two instructions refer to the accumulator: branch if the accumulator is equal to zero, or branch if it is not. But what they actually test is the state of a flag called the "Zero Flag". This flag is in fact a single bit of the Status Register. It takes the value of either ONE or ZERO depending upon whether the result of the last relevant operation was a zero or not (some operations, such as the store operations, do not affect the zero flag).

Thus if you test the zero flag after the instruction:

```
LDA #0
```

you will find that it has been set, since

the result of this instruction was zero. The two consecutive instructions:

```
LDA #0:BEQ somewhere
```

will result in a branch to the label "somewhere". To further emphasise the point we could loosely translate these two instructions as: "Load accumulator with zero. Branch to "somewhere" if the result of the last instruction was zero".

Suppose now that we sandwich a third instruction LDX #6 between these two to give:

```
LDA #0:LDX #6:BEQ somewhere
```

The result of this new sequence would be different, and the conditional branch would no longer be executed, since loading the X register also affects the zero flag. In this case we have loaded it with a non-zero value, thus unsetting the zero flag prior to the conditional branch instruction.

In the next issue we will look at some varied examples of branching, and introduce program loops, the equivalent of the FOR NEXT loop in Basic. But in the mean time, perhaps you can work out what the following piece of code does. What is the effect of executing JSR tester, and what is the broad equivalent of this in Basic?

```
.tester   LDA  &70
          BNE  notzero
.zero     LDA  #ASC("Y")
          JMP  write
.notzero  LDA  #ASC("N")
.write    JSR  oswrch
          RTS
```

BBC Micro-Specific Works

I.Birnbaum	Assembly Language Programming for the BBC Micro
B.Smith	BBC Micro Assembly Language

General Works

R.Zaks	Programming the 6502
L.Scanlon	6502 Software Design
L.Leventhal	6502 Assembly Language Programming

Table 3
Select Bibliography

28 BEEBUG Education

Product : Motion in Space
Supplier : Cambridge Micro Software
Shaftesbury Road,
Cambridge CB2 2RU.
Tel. (0223) 312393
Price : £20.13 inc. VAT

This is a brilliant - if overpriced - piece of software from the software division of Cambridge University Press. Again it could be used with most secondary age-ranges. There is a hidden facility for changing the level of difficulty, and the amount of detail in the instructions.

The suite simulates Newton's three laws of motion in two stages. First, you practice controlling an astronaut by the cursor keys, and learn what inertia really means. This is followed by a number of graded rescue and repair missions (treated as games) to test the interactive skills supposedly acquired in part one. There may be no sound, but you do get good smooth graphics, and simple and clear ideas for teaching well, tied to the underlying scientific principles. "Motion in Space" for the BBC B, B+ and Master does a useful, simple job appealingly and well.

UPDATE TO MIDAS - (BEEBUG Vol.6 No.1)

The author of Midas, Jonathan Temple, has provided the following modifications to the original program to extend and improve the recognition of different file types:

```
2530 IF E%<>L% T=7
2531 IF L%=0 T=1

2560 IF wise IF B%?4=&FF OR D%(F%)=87 OR D%=119 T=9
2570 IF F$="!BOOT" T=5 ELSE IF A%=255 T=6
2580 IF E%=&8019 OR E%=&801F OR E%=&8023 OR E%=&802B OR E%=&B823 T=8
```

Certain versions of Wordwise produce files that will not be recognised after these changes. To solve this, the directory for such files should be changed to 'w' or 'W'. Please note that it is impossible to write a program to correctly recognise every possible file. The original program will also respond to star commands.

will need a comprehensive third party front end to get the most out of the Archimedes' sound hardware.

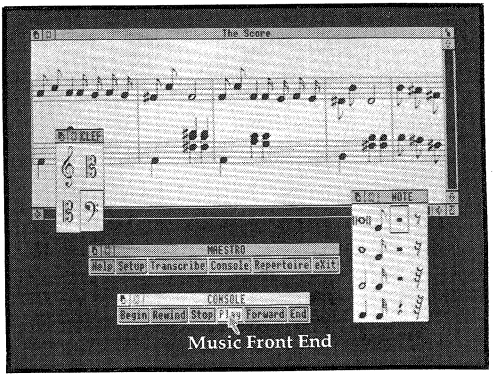
RELOCATABLE MODULES

With the A300 series machines Acorn have implemented a feature called Relocatable Modules. These are similar in many respects to sideways ROM or RAM in that they allow software packages to be incorporated as if they were a part of the machine's operating system. Modules, which may contain both star commands and help code, may be loaded from disc using:

*RMLOAD filename

No load address is required since all modules are located by the operating system software, and must be written in completely relocatable code. It is envisaged that programmers and software houses will produce a wide variety of modules for the new machine. Both the Machine Code Debugger and the Basic Editor supplied on the Welcome disc utilise this format.

Mirroring the notion of Relocatable Modules (see inset) at Basic level, Acorn have incorporated a modular programming facility into Basic V, in which the new



Basic directives LIBRARY or INSTALL may be used to load program modules into a reserved area of RAM. When an unrecognised function or procedure call is made from a program running in main memory, the resident library modules are also checked, and if the matching definition is found, it is executed as if it were in main memory. This facilitates the dynamic use of function/procedure libraries.

Perhaps taking a leaf out of Commodore's book, Basic V contains built in sprites. These are handled by a number of star commands, including *SLOAD and *SSAVE to load and save sprite images, and *SCHOOSE to select a particular sprite by name. The following sequence:

```
*SCHOOSE horse
PLOT &ED,640,512
```

will put the sprite called "horse" in the centre of the screen. A sprite definer utility is included on the Welcome disc.

Basic V also contains a full set of commands to support the mouse. These allow the user to write his own mouse-driven software. To show how simple this is, the ARM Basic User Guide provides a complete mouse-driven sketchpad program implemented in just 9 short lines. It makes use of the fact that the mouse pointer can be made to move automatically on screen to follow the movements of the user-controlled mouse.

ADFS

A300 series machines are supplied with the ADFS and ANFS as standard. Neither DFS nor CFS are provided; but mercifully - though clearly out of necessity - the ADFS has been completely rewritten. The very poor performance of "Hugo's" ADFS floppy disc code as implemented on the Master is now but a distant memory. And Acorn have complemented their 640K double sided format with an 800K mode which is said to be considerably faster. All this is completely invisible to the user. He merely decides which size of disc he requires prior to formatting, and in all subsequent use the ADFS recognises the disc size, and behaves accordingly. It is also claimed that the ADFS now takes less than 4 millennia to perceive that it cannot read a particular disc.

Acorn have also added one or two other nice refinements to their ARM ADFS (though there is still no *WIPE). These include automatic date-stamping of all files, and the assignment in the disc catalogue of file type. Thus if you perform *EX on a Basic file, it will not only give length and disc sector information, but will also print the word "Basic", and the time and date that the file was created.

The ADFS also incorporates a so-called "user root directory" option. This allows

the user to define a particular path from the root directory, and to subsequently access it from anywhere on the disc using "&", just as "@" is used to specify the currently selected directory. Unfortunately neither the library directory assignment nor the user root directory are stored on the disc itself, so that every time that you power up, or change a disc, you must tell the system all over again how you wish these assignments to be made.

ARM ASSEMBLER

The way in which the ARM assembler is implemented closely follows the Beeb's 6502 assembler. Even the standard operating system calls have been retained, although when these are called from Basic a new 'SYS' call is used rather than a CALL to the appropriate operating system address. Of course the assembler instructions themselves are very different from the 6502 set. Moreover the notion of "reduced instruction set" immediately conveys the wrong idea about ARM assembly language. There is no question of this being a REDUCED instruction set in the normal understanding of the word. True, there are only some 20 basic instruction types. But each of these is implemented conditionally, instantly multiplying the number of available instructions by a factor of 16. Thus while:

```
MOV R1,#&FF
```

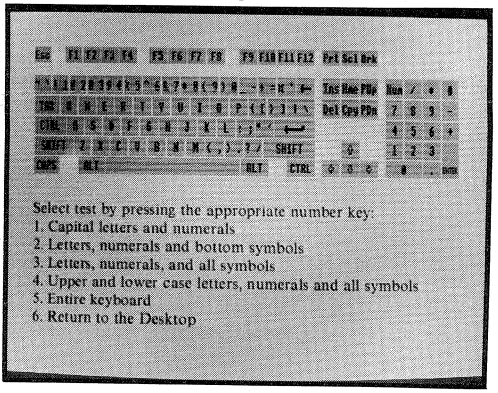
will indeed load register R1 with the value &FF, the load instruction LDR may also take the form:

```
LDREQ R1,[R2,R3]
```

This will load R1 with the contents of the address given by adding the contents of registers R2 and R3. This load is carried out conditionally on the zero flag being set (This is what the EQ following the LDR signifies). Programmers of the 8 bit 6502 should also pause to reflect on the pure magic of having registers as wide as the address bus - no more of this 6502 "low byte first" nonsense!

As we briefly mentioned last month, the ARM processor contains a powerful piece of hardware called a barrel shifter, which, within a single cycle, can shift or rotate a 32 bit word by up to 31 places. The shift and rotate instructions on the old 6502 could only shift one bit at a time. Moreover the shift and rotate

Keyboard Layout Showing One of the Available Fonts



operations on the ARM can be incorporated into other instructions. To illustrate this, consider the store operation:

```
STREQ R1,[R2,R3,shift#16]
```

The effect of this instruction, which is conditional on the zero flag, is to store the contents of register R1 at the address given by shifting the contents of R3 by 16 places, and adding the result to R2.

The recent addition of a Booth's multiplier to the 2 micron version of the ARM has made possible the implementation of two powerful multiply instructions MUL and MLA. The first gives a straight multiply of the contents of any two registers, where the result should not exceed 32 bits in length. The second performs a combined multiply and add operation. Thus:

```
MLANE R1,R2,R3,R4
```

will, unless the zero flag is set, perform:

```
R1:=R2xR3+R4
```

where "R1:=" means "set R1 equal to". Again both are conditional instructions, and like most instructions, may take an additional "S" parameter which means "set the flags on the result of this operation". And there is much more besides. ARM machine code is in reality a complex and sophisticated programming language; matched only by the complexity and sophistication of the Archimedes itself. Acorn have launched a truly spectacular computer, and we shall be following its progress with extreme interest in the coming months.

WIN ARCHIMEDES

Acorn has generously agreed to provide a brand new Archimedes A305 system with colour monitor (worth over £1100) as the prize in our competition to herald the launch of Archimedes. This is a tremendously exciting prize for the lucky winner, and we hope that many members will participate.

With the Acorn Archimedes being so fast, the theme of our competition is speed. You will need to write a program, as specified below, entirely in BBC Basic (no OSBYTE/FX statements or CALLs, no assembler or machine code). The program should be capable of running on a 32k BBC Model B, B+, Master 128 or Compact. To ensure compatibility with all the above machines the minimum PAGE setting allowed is &E00 (programs may naturally reside here or be moved down to this address), but your program must not make any use of shadow RAM or second processors.

THE PROGRAM

The task is to solve "Alphametics", which are alphabetic sums where each letter represents a unique digit. Thus:

```
  CROSS      represents  96233
  ROADS                  62513
 DANGER                 158746
```

The above example adds two lines to give the answer. However, the program must be able to cope with addition sums of up to ten lines. This example has three:

```
   HORSE
   OFTHE
    YEAR
  HARVEY
```

The winning program will be determined by its performance (standardised for machine type) in solving a series of Alphametics selected by the judges. The program should assume that only legitimate Alphametics will be entered (e.g. no more than ten different letters), and should ask if zeros are allowed or not. A legitimate Alphametic may have no solution, one solution, or more than one solution. Your program should include TIME statements, and display each solution and the time from starting to finding that solution. The program should also display the time

when it terminates having determined that no more solutions (if any) exist.

The programs will be judged entirely on the basis of speed, and programming style will not therefore feature in the judging. However, the winner must be able to supply a version suitable for publication (with notes on its operation) if requested.

BEEBUG will be the sole judges as to whether an entry conforms to the spirit of the competition, and sole judges when determining the winner. The judges decision will be final. No BEEBUG staff or their relatives are eligible to enter.

SEND A POSTCARD ONLY in the first instance with your name, address, daytime phone number, and membership number. Write clearly the time (in seconds) to find the first solution (if any) to the second of the two examples given, and the total running time of the program in this case. You must also state the system you used when running the program.

Those with the shortest times will then be asked to submit their programs on tape or disc for further testing by the judges to determine the overall winner. Send your entries (by 25th August 1987) to:

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

BEEBUG WORKSHOP

Rounding Errors

Although it is easy to believe that computers perform calculations accurately and reliably, the reality is in fact quite different as Surac explains.

THE PROBLEM

To the scientist and engineer, the computer is a tool just like a calculator, micrometer, calipers etc. In the same way the computer user must be able to trust the output that his computer gives. Apart from errors which arise from mistakes in the program, there are errors which are generated by the computer itself. We have been told that computers do not make mistakes; this is only half the truth. Computers process numeric values in ways which lead to imprecision. It is therefore essential that the programmer should be aware of these inaccuracies, and should know how they can be minimised.

To whet your appetite you may care to test the results when you program your computer to give the answer to the simple sum:

$$\frac{(1{,}000{,}001-1{,}000{,}000)}{0.0000001}$$

which as you can see has the result 10,000,000.

We also know that the sum:

$$\frac{1{,}000{,}001}{0.0000001} - \frac{1{,}000{,}000}{0.0000001}$$

should give the same answer. So try the following program which calculates the answer to the two expressions above:

```
10 A=10000001:B=10000000
20 C=0.0000001
30 D=A/C-B/C:E=(A-B)/C
40 PRINT D'E
```

You may be surprised to get the following result:

```
 10027008
 10000000
```

This highlights the fact that the difference in results depends on the chosen method of solution. One needs constantly to be aware of that fact to avoid errors.

ACCUMULATION OF ERRORS

Errors similar to that above can become alarmingly significant if a formula needs to be applied repeatedly to obtain a result. For example, the loop below will never end, although it appears that it should stop when A=2.

```
A=0:REPEAT:A=A+0.1:PRINT A:UNTIL A=2
```

Try the following statement typed in directly - you should get the message "OK", but you don't!

```
IF 0.7*0.7=0.49 PRINT"OK" ELSE PRINT"!"
```

To see why this doesn't work as expected try:

```
PRINT 0.7*0.7
```

This displays 0.49, so where is the problem? To force the computer to show us the difference between 0.7*0.7 and 0.49 perform the following:

```
PRINT 0.7*0.7-0.49
```

and now at last we can see that we do not get the ZERO expected, but:

```
-1.16415322E-10.
```

What has happened is that the computer, in performing a test for equality, subtracts the result of 0.7*0.7 from 0.49, and tests the result against zero. As the result is not zero the ELSE part of the IF statement is performed.

The next program is designed to add 100 pennies to the bank, after which it adds 1 to the number of pounds to keep track of where it should be. After each addition of a pound, it displays what is in the bank and the number of pounds. You will find that the two are not keeping pace - why?

```
10 penny=.01:pounds%=0:bank=0
20 REPEAT
30 FOR I=1 TO 100:bank=bank+penny:NEXT
40   pounds%=pounds%+1
50   PRINT bank, pounds%
60 UNTIL FALSE
```

THE REASONS FOR THE ERRORS

As we all know, there are many fractions that are impossible to represent

exactly as a decimal. A glance at table 1 will show how numbers that are representable EXACTLY in base 10 are not necessarily (although they are sometimes) exactly representable in binary.

Decimal Fraction	Decimal	Binary
1/2	0.5	0.1
2/5	0.4	0.01100110011...
1/4	0.25	0.01
1/10	0.1	0.0001100110011...
1/32	0.03125	0.00001

Table 1

Inaccuracies arise for two reasons, (i) when numbers that have no exact binary representation are used in calculations, and (ii) when the finite number of bits used for storage is insufficient to represent the number exactly.

In order to demonstrate the problem, I have chosen as examples numbers that can be represented exactly in base 10, but not in binary. If you were to rewrite the examples using a decimal number that COULD be represented exactly in binary, say 0.5, then the problems would disappear. For example try:

```
IF 0.5*0.5=0.25 PRINT"OK" ELSE PRINT"!"
```

and you will see "OK" confirming that all is as expected.

The errors are compounded when repeated additions, multiplications, etc are performed on numbers that cannot be represented exactly, as the two looping programs earlier show.

REDUCING ERRORS

To minimise errors you should always work with integers if at all possible. For example, handle monetary amounts in pennies, and only divide by 100 to convert to pounds at the last stage in the process. We are fortunate in that BBC Basic allows integers with up to 10 significant figures. Most other Basics only allow integers in the range -32767 to +32767.

Loop variables should always be integers; if non-integers are required then scale the number inside the loop rather than let the errors accumulate. For example, compare the output from the following two routines:

```
FOR I=0 TO 10 STEP 0.1:PRINT I:NEXT
FOR I%=0 TO 100:I=I%/10:PRINT I:NEXT
```

The first routine doesn't even perform the loop the required number of times! Notice how it doesn't reach the final value of 10, and note also the values output.

Should you have no alternative than to produce a result with rounding errors, you will find the following function useful to produce the number n to p decimal places:

```
1000 DEFFNround(n,p)=INT(10^p*n+0.5)/10^p
```

Finally the following program will allow you to experiment with various decimal numbers and their binary equivalents, and see how the accuracy

The program prompts you to enter any decimal fraction between 0 and 1 (e.g. 0.0127). The program then calculates and displays the nearest binary equivalent to the specified number using 4, 8, 16 and 32 bits in turn. In each case, the program displays the true decimal value of the binary equivalent so that the degree of error may be seen.

DECIMAL TO BINARY

```
100 MODE3
110 DIM b$(32):@%=&00020A0A
120 RESTORE:REPEAT
130 INPUT"Decimal fraction between 0 an
      d 1 ",d
140 UNTIL d>=0 AND d<1
150 dec=d
160 FOR i=1 TO 32
170 dec=dec*2
180 IF dec>=1 b$(i)="1":dec=dec-1 ELSE
      b$(i)="0"
190 NEXT
200 PRINT"Decimal"TAB(14)"Binary";
210 PRINT TAB(50)"No of  Actual"
220 PRINT"fraction"TAB(14)"Fraction";
230 PRINT TAB(50)"bits   value"
240 PRINT"--------"TAB(14)"--------";
250 PRINT TAB(50)"-----  ------"
260 DATA 4,8,16,32,0
270 REPEAT:READ n:IF n=0 THEN GOTO 120
280 PRINT d;TAB(14);
290 frac=1:tot=0
300 FOR i=1 TO n
310 PRINT b$(i);
320 frac=frac/2
330 IF b$(i)="1" tot=tot+frac
340 NEXT
350 PRINTTAB(50);n;TAB(55);tot
360 UNTIL FALSE
370 END
```

POSTBAG

POSTBAG

Bigger Sums in Wordwise

It was interesting to read the letter (BEEBUG Vol.6 No.1) on 'Summing up Wordwise'. I too require similar processing from Wordwise Plus, but my needs exceed the £655.35 limit. The program below can be used similarly to that of Mr Ridgewell when entered into a segment. However, this program will allow higher totals to be processed (£65535.00). When entering amounts into the text area, it will not matter whether or not there is text before or after the amount, as the program looks for and processes the characters after a "£" up to a "." and then two more characters.

```
X%=0
Y%=0
SELECT TEXT
CURSOR TOP
.top
X$=""
Y$=""
FIND "£"
IF EOT THEN GOTO end
CURSOR RIGHT
REPEAT
A$=GCT$
IF A$<>" " AND A$<>"." THEN
    X$=X$+A$
UNTIL A$="."
DOTHIS
A$=GCT$
IF A$<>" " THEN Y$=Y$+A$
TIMES 2
X%=X%+VALX$
Y%=Y%+VALY$
GOTO top
.end
X%=X%+Y%DIV100
Y%=Y%MOD100
A$="Total = "+STR$X%+"."
IF Y%<10 THEN A$=A$+"0"
TYPE A$+STR$Y%
*FX138,0,0
END
```

Mike Holdaway

Auto-Run Basic

With reference to the article by Dr.R.D.Bagnall in Vol.5 No.10, I would like to comment as follows. The article says that, " it appears that the state of a program can only be saved satisfactorily if it is not executing a REPEAT-UNTIL or FOR-NEXT loop".

It is possible to save the program during the execution of such loops (and GOSUB-RETURN loops) if more Basic zero page workspace is saved. This is because the loop counters are stored in memory at &24 to &26. It is therefore necessary to save &00 to &28 (the assembler OPT flag).

Unfortunately I cannot make the Auto-Run method work with functions and procedures even after many attempts.

R.J.Titmus

This is an interesting area for further investigation. The problem with functions and procedures is likely to arise from their use of the stack (below HIMEM) to hold appropriate return addresses. However, although we have experimented a little with this, the results are so far inconclusive. Maybe the answer is to save the whole of user RAM from &00 to &7FFF, including the screen display which might then be automatically reproduced!

Recursive Comment

I have some comments about the recursive binary search procedure in the Workshop for Vol.5 No.9. It IS necessary to declare mid% as LOCAL, but this is not essential for the variables done% and temp (as their values are irrelevant after recursively calling the function).

Secondly, the following function is faster, shorter and more elegant, and needs no explicitly declared LOCAL variables.

```
DEF FNsrch(I%,J%,match)
IF J%-I%<=1 THEN =-1:
    REM no match exists
K%=(I%+J%) DIV 2: T=A(K%)
IF T=match THEN =K%:
    REM found
IF  T<match THEN I%=K%
    ELSE J%=K%
=FNsrch(I%,J%,match)
```

To search from P% to Q% for 'match', a suitable call would be:

```
Z%=FNsrch(P%-1,Q%+1,match)
```

This function is nearly twice as fast as the original FNchop.

A.G.Rimmer

Our thanks to Mr Rimmer for his comments and suggestions. The correct use of LOCAL can be crucial in recursive procedures to ensure that values are properly preserved at the various levels of recursion. Variables referenced before any recursive call should be so referenced, while those after need not.

B

HINTS and tips HINTS and tips HINTS and tips HINTS and tips HINTS and tip

Inter-Word and Long Commands

Should you require more embedded commands than can be held in a single control code line, then splitting these into two lines prints a space between each. This can be overcome as follows.

Enter the embedded command menu, and enter the first 6 or 7 codes:
eg: 135,133,133,133,133,133
Exit the menu, type space, and re-enter the menu, and type the next sequence:
130,133,133,133,133,136
Exit the menu, type space, and re-enter the menu, and type the next sequence:
130,133,133,133,133,136
This can be continued as long as the maximum space for control codes per line is not exceeded. (Beware: you get no warning of exceeding the limit, and cannot recover if you do!).

Chris Wragg

Word Processing Basic Programs

Following the hint in last month's magazine an alternative method has been proposed.

This method has been tried on Wordwise and View. In Inter-Word you must place markers round the program and save it using option 3, not option 1 which saves the document configuration as well as the text. This method will doubtless also work on most other word processors.

Place the word AUTO by itself as the first line of your program in the word processor. Do NOT allocate line numbers to any of your program lines.

Once the program has been completely entered and the file saved, enter Basic and type:

```
*EXEC filename
```

This will put your program into Basic complete with line numbers. Remember to press Escape to get out of the "AUTO" mode at the end. You will need to go back through your program to deal with GOTOs etc. in those iso- lated occasions where you had to use them!

Dates in Wordwise

For those of us not using a Master but requiring documents to be dated, it is sometimes frustrating to find the previously-used date incorporated in the header or footer of the document. To overcome this, put:

```
<f1>TI50<space>1987
```

where the date should go, and save the document. Now when you forget to add <f2><date> after the 50 you will get "bad operator" when you preview or print the file, as a reminder.

T.K.Boyd

Two Programs in Memory at Once

The following instructions show you how to hold two (or more) Basic programs in memory at the same time. Enter:

```
Y%=PAGE
```

(this saves your current page setting). Then type in (or load) a Basic program, followed by:

```
Z%=((TOP DIV &100)+1)*
   &100
PAGE=Z%
```

This resets PAGE (the address where Basic programs start), to the next available block. Now type NEW, and type (or load) in the next program, followed by:

```
*KEY0 PAGE=Y%|M
*KEY1 PAGE=Z%|M
```

To access the first program you will need to press f0. Similarly, the second program is accessed by pressing f1. Each program may be run, loaded or saved in the normal way.

Max Christian

Self-Destruct Function Keys

When a function key definition in a program is to be deleted immediately after it has been called (as in a move-down routine), it is only necessary to end the definition with *K.#|M where # is the number of the key programmed. This avoids the dreaded "Bad key" message often encountered on subsequent re-definition.

D.P.Dyer

Crosswords with Spellmaster

The following !BOOT file held on disc will satisfy most crossword addicts by entering Spellmaster, loading the "User Dictionary", and programming the function keys:

```
*BUILD !BOOT <Ret.>
1 *KEY0 "*ANA." <Ret.>
2 *KEY1 "*BRO." <Ret.>
3 *KEY2 "*CHE." <Ret.>
4 *KEY3 "*CRO." <Ret.>
5 *KEY4 "*DLOAD" <Ret.>
6 *KEY5 "*DSAVE" <Ret.>
7 *KEY6 "*FUZ." <Ret.>
8 *SPELL <Ret.>
9 *DLOAD <Ret.>
10 <Escape>
```

Don't forget *OPT4,3 to enable auto-booting when you press Shift-Break.

John L. Taylor

BEEBUG MEMBERSHIP

Send applications for membership, membership renewals, membership queries and orders for back issues to the address shown. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank.

MEMBERSHIP SUBSCRIPTION RATES

£6.90 - 6 months (5 issues) UK ONLY

£12.90 - 1 year (10 issues) UK, B.F.P.O., Channel Islands

£19.00 - Rest of Europe

£26.00 - Americas & Africa

£23.50 - Middle East

£28.00 - Elsewhere

BACK ISSUES (Members only)

VOLUME	SINGLE ISSUES	VOLUME SETS (10 ISSUES)
1	40p	single issues 7, 9 & 10 only
2	50p	£3.50
3	70p	£5.50
4	90p	£7.50
5	£1.20	£10.50
6	£1.30	–

DISCOUNT: Any 10 or more issues, deduct £1.50 from the total single issue price. 20 or more deduct £3.00, 30 or more deduct £4.50, etc.

Please add the cost of post and packing as shown:

DESTINATION	FIRST ISSUE	EACH SUBSEQUENT ISSUE
UK, BFPO + CH. IS.	40p	20p
Europe + Eire	75p	45p
Elsewhere	£2	85p

All overseas items are sent airmail (please send a sterling cheque). 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 that require an invoice. Note that there is no VAT on magazines.

Back issues are for members only, so it is ESSENTIAL to quote your membership number with your order.

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

St. Albans (0727) 40303
Manned Mon-Fri 9am-5.00pm

(24hr Answerphone Service for Access/Visa orders and subscriptions)

If you require members discount it is essential to quote your membership number and claim the discount when ordering.

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Editorial Assistant: Kristina Lucas
Production Assistant: Yolanda Turuelo
Membership Secretary: Sara Anketell-Jones
Editorial Consultant: Lee Calcraft

Additional thanks are due to Sheridan Williams, Adrian Calcraft, John Yale, Tim Powys-Lybbe, and Dave Somers.

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Limited.

BEEBUG Ltd © 1987.

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 £40 per page, but please give us warning of anything substantial that you intend to write. A leaflet, 'Notes of Guidance for Contributors', is available on receipt of an A5 (or larger) SAE.

In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "View", or other means, but please ensure an adequate written description of your contribution is also included. If you use cassette, please include a backup copy at 300 baud.

In all communications, please quote your membership number, and enclose a SAE if you require a reply.

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

Produced by Head Office Design (0782) 286375 ISSN – 0263 – 7561

Magazine Cassette/Disc

JULY 1987 CASSETTE/DISC CONTENTS

QUADRIC SURFACES — a program for the display of three dimensional surfaces on the screen which employs a technique for giving added depth to 3D displays.

PERSONAL ADDRESS BOOK — an easy-to-use address book allowing up to 75 entries for each letter of the alphabet with a total of 2000 entries in all.

THE MASTER SERIES

TURBO ADFS MENU — written in machine code, this provides rapid traversal and menu selection for any ADFS system (model B, Master and Compact).

BARRY CHRISTIE VISUALS — a very smooth giant scroller from this master of the visual effect.

MODE 7 EDITOR — a highly useful editor for all those mode 7 screens.

BEEBUG WORKSHOP — demonstration of the effect of rounding errors.

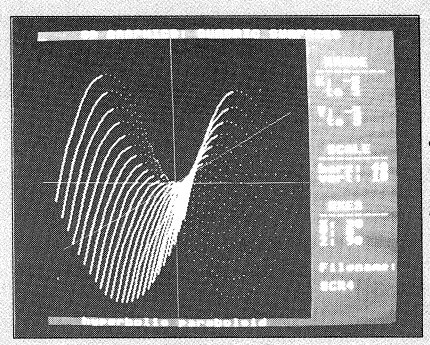
FIRST COURSE — three complete example programs showing the use of fill routines.

EXPLORING ASSEMBLER (PART 2) — another example from this new series on assembler programming.

HOW TO WRITE AN ADVENTURE GAME — part one and part two combined to provide a complete working adventure game called 'Jungle Adventure'.

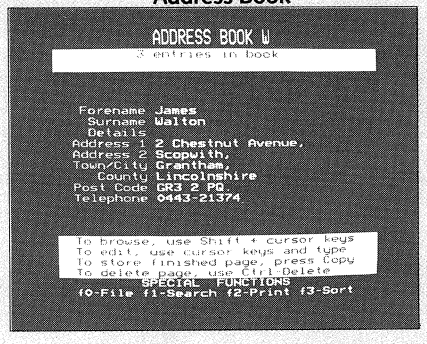
EXTRA FEATURES THIS MONTH

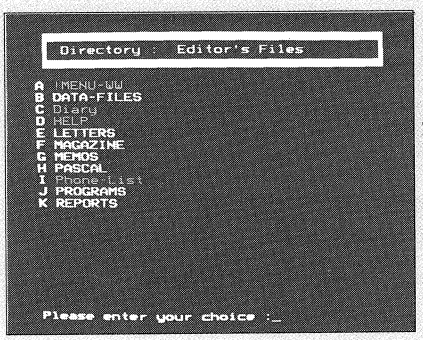
MAGSCAN — data for this issue of BEEBUG (Vol. 6 No. 2).



Quadric Surfaces

Address Book





Turbo ADFS Menu

All this for £3.00 (cass) £4.75 ($5\frac{1}{4}$" disc) £5.75 ($3\frac{1}{2}$" disc) + 50p p&p.
Back issues ($5\frac{1}{4}$" disc since Vol.3 No.1, cass since Vol.1 No.10) available at the same prices.

Subscription rates (UK only)	*$5\frac{1}{4}$" Disc*	*$3\frac{1}{2}$" Disc*	*Cass*	*Subscription rates* (Overseas)	*$5\frac{1}{4}$" Disc*	*$3\frac{1}{2}$" Disc*	*Cass*
6 Months (5 issues)	£25.50	£29.50	£17.00	6 months (5 issues)	£30.00	£34.00	£20.00
12 Months (10 issues)	£50.00	£58.00	£33.00	12 months (10 issues)	£56.00	£64.00	£39.00

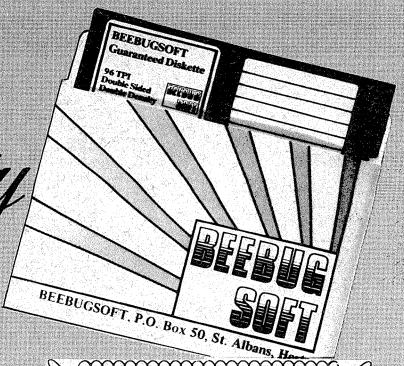
Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to $5\frac{1}{4}$" disc subscription on receipt of £1.70 per issue of the subscription left to run. ($3\frac{1}{2}$" disc £2 per issue)

All subscriptions and individual orders to:
BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts. AL1 1EX.

The Ultimate in Quality & Reliability

BEEBUG DISCS



Our blank discs are produced for Beebug by one of the worlds leading disc manufacturers in a special dust free environment. Every single disc is then individually tested for data reliability and any sub-standard units are instantly rejected.

Our Guarantee

We confidently offer a lifetime data guarantee and will replace any disc with which you encounter problems. We have found that the standard of quality control at the factory makes this necessity very rare.

40 Track Single Sided Double Density

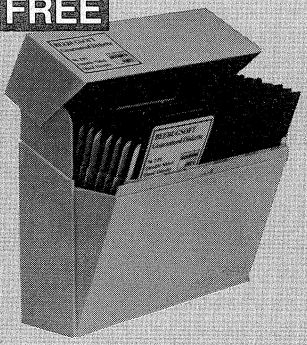
	Price	Members Price	Order Code
10	£9.90	£9.40	0657
25	£23.00	£21.85	0661
50	£48.00	£45.60	0665

80 Track Double Sided Quad Density

	Price	Members Price	Order Code
10	£10.90	£10.35	0660
25	£29.00	£27.55	0664
50	£53.00	£50.35	0668

Can you afford to trust your data to anything less reliable? Many unbranded discs on the market are produced from low quality media rejected by the major manufacturers.

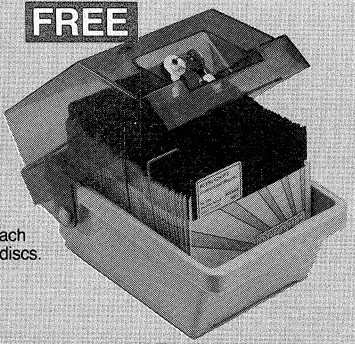
FREE



Memorex Plastic Library Case

Supplied with each purchase of 10 discs.

FREE



New LOCKABLE Storage Box

(Normal Price £13.80)

With special carrying handle with each purchase of 25 or 50.

All Prices Include VAT.

UK Carriage 10 £1.00, 25/50 £3.75.
Overseas send same price including UK carriage & VAT (To cover extra postage).

Official Orders Welcome.

Hotline for Access/Visa telephone orders 0727 40303 (24 hours).

Name_____

Address_____

Please send me ____ (qty) ____ (stock code) at £____ (unit price) ____ plus £____ (postage).

I enclose a cheque value £____

Please debit my Access/Visa with £____

Card No. | | | | | | | | | | | | | | | | |

Expiry ____/____

Membership No. (To claim members price) ____

BEEBUG

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

Telephone: 0727 40303