Beebug August/September 1985

CABLE TO USER PORT £1.20

j i h g f e d c b a Z Y X W V U T S R Q P O N M L K J I H G F E D C B A

BEEBUG

1Ω

10mH

TL497

74374

1KΩ

1KΩ

20KΩ

1.2KΩ

100µF

330pF

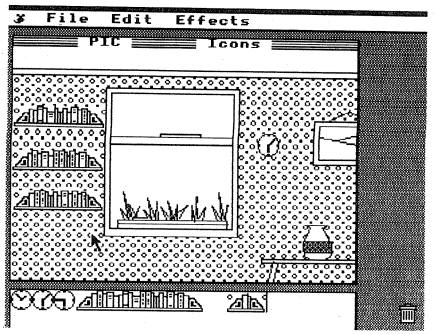
EPROM PROGRAMMER PROJECT

for the BBC micro

4.7KΩ

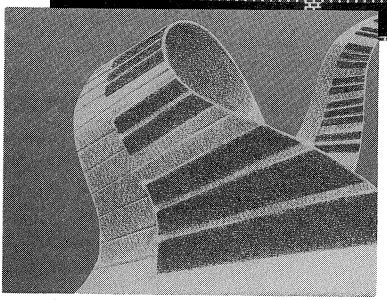
j i h g f e d c b a Z Y X W V U T S R Q P O N M L K J I H G F E D C B A

CABLE TO PRINTER PORT



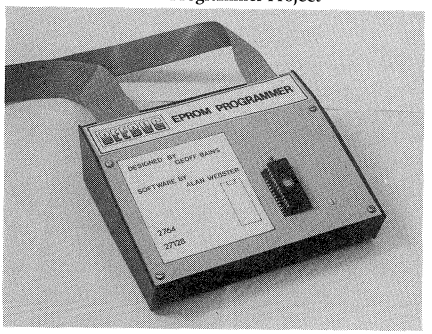
The AMX Mouse Grows Up

Builder Bob



Creative Sound Reviewed

EPROM Programmer Project



BEEBUG

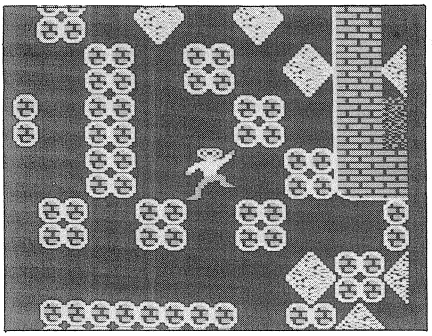
August/September 1985

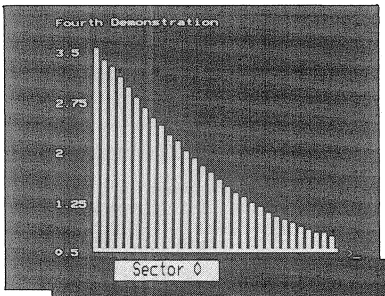
GENERAL CONTENTS

PROGRAMS

HINTS, TIPS & INFO

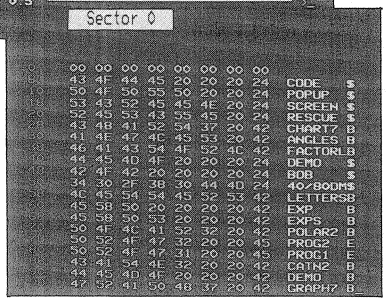
DAVID BARNES LOW



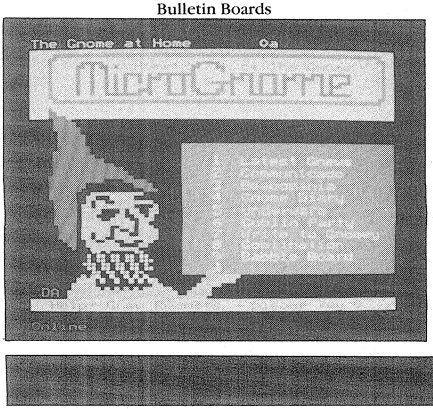


Drawing Simple Bar-charts

Disc Catalogue



Bulletin Boards



EDITORIAL JOTTINGS

Since the start of Volume 4 of BEEBUG, we have made some changes to the layout and presentation of articles and programs in the magazine. One of the consequences of this is that our standard column width is now 42 characters. We have taken advantage of this to list all our programs in a 40 column width. This means that when you enter a program listed in the magazine into your Beeb, and then list it in 40 column mode on the screen, all the line breaks should match up exactly with the magazine listing, making checking and location of any typing errors much easier.

There are two points to note. All our programs are listed in the magazine with a LISTO1 format, and a program must be listed in this format on the screen if it is to match up. The other consequence of this approach is that you will find several instances in most programs where the end of a line of Basic overlaps onto the next physical line by a single character, which is only too easily missed. We could avoid this, as we used to do previously, but only by dropping a strict adherence to the 40 column format. We believe this small cost is worthwhile.

Pursuing our policy of providing more special offers on selected hardware and software, we are now able to offer Oxford Pascal (reviewed BEEBUG Vol. 4 No.3) at a discount to members. With the success of Wordwise Plus, BEEBUGSOFT have released a new complementary product called 'Wordease' which provides 15 powerful and useful utilities to enhance the use of Wordwise Plus. Details of both these items, and other offers to members, are included with this issue.

This month's magazine cassette/disc contains three extra items. There is the revised version of the Polar Curves program first published in BEEBUG Vol. 4 No. 1 and referred to in this month's BEEBUG Workshop. Following the Extended 65C02 Assembler featured in the same recent issue, we are now included a disassembler (by Nick Hudson) which also handles the full 65C02 instruction set. Thirdly, we have put in a full-feature machine code game by P. Oliver, called Rescue Mission. See the back cover for details of this and all the other programs.

We have decided this month to award three prizes of £10 for the hints by Douglas Siviter, Richard Sterry and Dave Brincombe. We always welcome your hints and tips, so keep them coming.

PROGRAM CLASSIFICATION

All programs in the magazine, and on the magazine cassette/disc, are marked with the symbols shown below. An uncrossed symbol indicates full working, a single line through a symbol shows partial working (normally some modifications will be required), and a cross through a symbol indicates a program that will not work on that system. We also have a symbol for the recent model B+.

Basic 1	I	Electron	
Basic II	II	Disc	
Tube		Cassette	
Model B+			

POSTBAG

POSTBAG

DOMESTIC ACCOUNTS DATED

I have modified the Domestic Accounts program (BEEBUG Vol.2 No.10, Vol.3 No.6) so that different dates can be saved with the file (lines 2651 and 2652). I have also taken the opportunity to improve (in my view) the presentation of data.

```
 200 IF R=65 CLEAR:R=0:DIM
BCF%(13):pad$="     = = ":P
ROCload:PROCmonths
2110 OPNBAL=0:pad$="     =
= "
2720 MON$(I%)="    ="+A$+"=
":NEXT
3880 READ A$:PRINT A$:

2651 PROCcolours
2652 PROCdate
```

These changes make the presentation more readable and ensure the correct date goes on the file

Ian Kidd

LEAP OF THE CENTURY

I read with interest the article and subsequent letters on the calendar program (BEEBUG Vol.3 No.7). I thought to myself "It isn't going to work because century years are NOT leap years." Before writing to you I decided to check my facts.....and all is well!

A year is actually 365.2425 days (1460.97 after 4 years). A year with leap-year compensation is actually 365.25 days (1461 after 4 years. This is still 0.03 days out! After 400 years the figures are 146097 and 146100; 3 years out exactly. This is compensated for by making all century years that are NOT divisible by 400 into non leap years. Thus 1600 was a leap year but 1700, 1800 and 1900 were not. I have used the term 'century' for convenience, but I would comment that the year 2000 is not the start of a new century as many ignoramuses (ignorami?) would have us believe.

Stephen Kirby

SHEILA IN TROUBLE

While trying to interface a number of Beeb's together using a program that directly addresses &FE08, 9 and 10 (SHEILA), one of my friend's computers would not work. On investigating it was found that his IC7 interface chip was different. Instead of a Ferranti 2C199 it was a 'special' Acorn chip. These, I understand, were installed in a number of Beebs about 18 months ago. Changing the bad chip then allowed direct and correct access to SHEILA.

B.D.Cocksedge

Acorn acknowledges that there are two sources of the chip in question and that in certain circumstances they may behave differently. For RS423 working, bits 3, 4 and 5 of the control register in the Serial Processor IC are set to the receive baud rate. This may be done using *FX7,X. For cassette working these three bits should be set to zero, as the Serial Processor does this automatically. If *FX7 is used, incorrectly, to set the receive baud rate in these circumstances, the function of the Serial Processor is undefined and the two versions behave differently. Acorn recommends that software written to replace the CFS (and directly access the relevant SHEILA locations) should always set these serial bits to zero, which may be done using *FX7,8.

PROCEEDING DYNAMICALLY

The program for dynamic loading of functions and procedures (BEEBUG Vol.3 No.2) allows you to squeeze bigger programs into the Beeb's limited memory. Recently I used the technique to support a program with a dozen overlays. In doing this I discovered that it was still possible to obtain the very 'No Room' error that the program was designed to avoid. This can be corrected by adding

```
2275 AND #&7F \Remove 'hid
ding' bit
```

Once this is added, two or more overlays may be swapped indefinitely.

Chris Reynolds

We have checked this out and it works. In fact, it would appear that this line was accidently left out of the original program at some stage. This very useful program is also part of Discmaster produced by BEEBUGSOFT, and this amendment applies also to that version.

News News News News News News.....

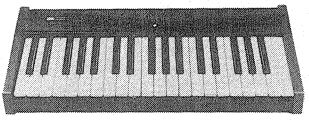
LOGO GOES ON

Logo Software Ltd. has issued a version of its Logo (reviewed Vol.3 No.10) for Econet systems. There is a licence fee of £40 per network and manuals cost £15 each. Minimum package is a licence and four manuals at £100. LSL is on 01-891 0989.

Meanwhile, Logotron has released its promised sprite board for £179.95. The Advanced Logo to make use of the sprite board and provide a host of other goodies (control Logo and more primitives) costs £15. Unfortunately the board is only controllable from Logo at present though plans are under way to produce some Basic control. There are plans also to produce some Logo applications software before the end of the year. This will include a spreadsheet and a maze-solving program. Logotron is on 01-352 1088.

TOOLBOX 2

Members with old Beebs and long memories will remember BBC Soft's early release, 'Toolbox'. A follow up package called (none too originally) Toolbox 2 is now available. Like the first, Toolbox 2 contains a whole host of utility programs to help you with Basic, assembler, graphics, filing, and other bits and bobs. All the programs can be used on their own or as subroutines in your own works. Toolbox 2 cost £9.95 for the tape and £10.95 for the accompanying book. Details from BBC Soft on 01-927 4518.



PLAY IT AGAIN ROM

LVL's musical keyboard for the Beeb has gained some new ROM driving software. The Echo keyboard is a 36 note miniature keyboard that connects to the User Port. The Echo ROM provides a music 'language' enabling you to store and recall played pieces and alter the envelopes, etc. of the sounds. The Echo ROM costs £29.95 from LVL on 0602-394000.

LOOKING AHEAD

Operation Caretaker is the unlikely name for a useful package for cassette users from Global Software. The package comprises both a cassette head cleaner and demagnetizer, and another tape containing a program to help you align your heads correctly! £10.95 will buy you the two tapes complete with a screwdriver to do all the adjusting. Details from Global on 01-228 1360.

TWO-DISC ADVENTURE

Following Acornsoft's 'Acheton', Robico Software has produced a text-only adventure on two discs. With over 450 locations and long, 'atmospheric' descriptions, 'Enthar Seven' contains around 150K of text compressed onto the disc. The game has a science fiction setting and costs £17.95 from Robico on 0443-227354.

PIE IN THE SKY

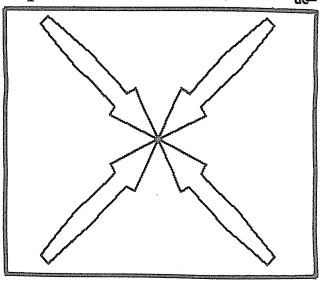
If you're looking for a more unusual use for your Beeb, what better than a satellite tracking program. AMSAT-UK have the goods in the form of Satpak. This package will help you predict satellite orbits and decode, display, and correlate data from the University of Surrey's amateur satellite. Further details from AMSAT-UK, 94 Herongate Road, Wanstead Park, NE12 5EQ.

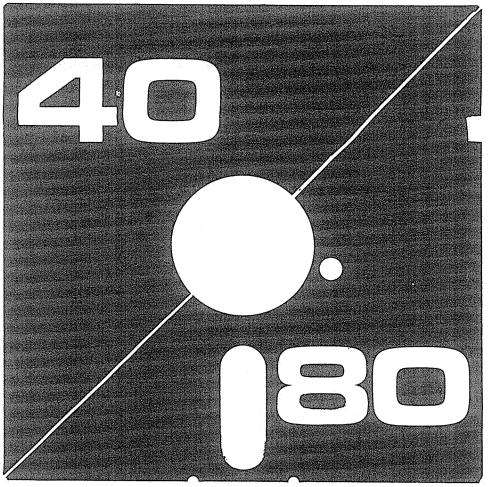
POLAR COMPETITION RESULTS

The Polar Curves competition prompted a good response from readers with many clever and original polar curve equations entered. However, the winner was selected as M. Jan Plaisier of Belgium for his arrows design. M. Plaisier's winning equation is remarkably simple:

```
SQR(ABS(TAN(theta*2)))AND-1
```

with a size of 200. The result is shown below. £50 will soon be winging its way to Mr. Plaisier.





DUAL FORMAT UTILITY

With two standards of disc drive (40 and 80 track) in use with the Beeb, creating a disc readable by both can only be useful. Steve Turnbull reveals the secrets involved and describes his simple-to-use 40/80 formatter.

Although a BBC micro with a disc drive is a powerful combination, problems can arise with the existence of both 40 and 80 track discs as these are not interchangeable. You may have trouble swapping software with your friends or at the local computer club, and software houses certainly don't relish producing two different disc versions of every program.

It is possible to purchase switchable 40/80 drives (at extra cost), and some (non Acorn) DFS ROMs can read both 40 and 80 track discs on an 80 track drive. Even so, many readers are likely to be limited to using just one standard of disc.

The solution is to use a 'dual-formatted' disc which can be read by both 40 and 80 track drives. The 40/80 utility described here will enable you to do just that, producing a dual 40/80 track disc. To use this utility you must have an 80 track drive, or a 40/80 switchable drive, (or both a 40 and an 80 track drive).

BASIC PRINCIPLES

It is possible to format an 80 track disc, partly as a 40 track disc and partly as an 80 track disc, with a common directory. The two areas cannot be interleaved which reduces the usable area, but with some judicious programming we can provide approximately 64K in each mode.

FORMATTING 40/80 DISCS

Load and run the program listed here, though make sure you have a copy safely stored away first. As soon as the 40/80 program is loaded, make sure you remove from the drive the disc on which it was stored. Now proceed as follows:

1. As prompted, specify the drives to be used for formatting at 40 and 80 tracks. If you only have an 80 track drive answer the same for both questions (the program automatically 'double steps' when creating the 40 track section).

2. Insert the disc to be formatted (it is best to use a previously unformatted disc) into the 40 track drive, (or the 80 if that's all you have), and press Return. The program first checks to see if there is any data on the disc.

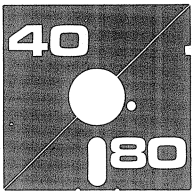
3. The 40 track section is formatted first, and then, as prompted, the 80 track section. Two dummy files are created (to 'hide' the unused tracks).

4. The disc is now dual-formatted and can be removed from the drive. Further discs can be similarly formatted.

SAVING FILES ON A 40/80 DISC

If you only have an 80 track drive which is not software or hardware switchable to 40 track (double-step) mode, then it is impossible to save files correctly on a dual-formatted disc. Using dual (or switchable) 40 and 80 drives, insert the disc into the 40 track drive and save the program. Now insert the disc into the other drive and save the program again using the SAME filename.

You must ALWAYS save a program on both parts of the disc at the same time to avoid catastrophic catalogue upsets. How-



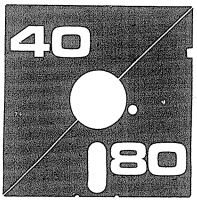
ever, you only need to delete a file once as this immediately changes the common catalogue. In fact, all the usual DFS commands will work correctly in 40 or 80 track mode, except *COMPACT.

USING SWITCHABLE 40/80 DRIVES

If you intend using a switchable 40/80 drive, switching first to 40 track mode and then to 80 track mode when formatting, you will need to modify two lines in the program. Replace 'drive40=drive80' in lines 2250 and 2260 with 'FALSE'. The modified program will not then function correctly on a single 80 track drive.

PROGRAM NOTES - see page 14.

```
  10 REM Dual 40/80 Disc Formatter
  20 REM Version B1.0
  30 REM Author Steve Turnbull
  40 REM BEEBUG Aug/Sept 1985
  50 REM Program subject to copyright
  60 :
 100 MODE7:HIMEM=HIMEM-&200
 110 ON ERROR MODE7:PROCe
 120 T$="40/80 Disc Maker"
 130 A$="S.D.Turnbull"
 140 PROCinitialize:PROCtitle
 150 ON ERROR PROCerror
 160 REPEAT PROCmake
 170 UNTIL NOT FNagain
 180 PROCquit:END
 190 :
1000 DEF PROCerror
1010 IF ERR=17 PROCerr:ENDPROC
1020 IF ERR>128 PROCfserr:ENDPROC
1030 DEF PROCe
1040 ON ERROR OFF:PROCquit
1050 REPORT:PRINT" at line ";ERL:END
1060 :
1070 DEF PROCquit:*FX15
1080 CLS:PROCcon:*FX4
1090 PROCdub(CHR$129+T$+CHR$135,8)
1100 PRINT:ENDPROC
1110 :
1120 DEF PROCcon:VDU23;&FF0B;0;0;0;:END
PROC
1130 :
1140 DEF PROCcoff:VDU23;11;0;0;0;:ENDPR
OC
1150 :
1160 DEF PROCdub(t$,Y%)
1170 PROCcen(CHR$141+t$,Y%)
1180 PROCcen(CHR$141+t$,Y%+1)
1190 ENDPROC
1200 :
1210 DEF PROCcen(t$,Y%)
1220 LOCAL X%:X%=18-LENt$DIV2
1230 PRINTTAB(X%,Y%)t$;
1240 ENDPROC
1250 :
1260 DEF PROCtitle:PROCcoff
1270 LOCAL Z%
1280 PROCdub(CHR$129+T$+CHR$135,7)
1290 PROCcen(CHR$130+A$+CHR$135,14)
1300 Z%=INKEY500:ENDPROC
1310 :
1320 DEF PROCinitialize:*FX4,1
1330 LOCAL E%
1340 err=0:cr$=CHR$13:*OPT
1350 READ errs:DIM e$(errs)
1360 FOR E%=0 TO errs-1
1370 READ e$(E%):NEXT
1380 error=TRUE:buffer=HIMEM
1390 osfile=&FFDD:osword=&FFF1
1400 oscli=&FFF7:osgbpb=&FFD1
1410 sectors=10:ctrl=&50:ENDPROC
1420 :
1430 DATA 10
1440 DATA Abort
1450 DATA Clock error
1460 DATA Late DMA
1470 DATA ID CRC error
1480 DATA Data CRC error
1490 DATA Drive not ready
1500 DATA Read-only disk
1510 DATA Track 0 not found
1520 DATA Write fault
1530 DATA Sector not found
1540 :
1550 DEF FNin(k$):PROCcon:LOCAL I%
1560 REPEAT I%=INSTR(k$,GET$)
1570 VDU-7*(I%=0):UNTIL I%<>0
1580 PROCcoff:=I%
1590 :
1600 DEF PROChead(t$):VDU26,12
1610 PROCdub(CHR$133+t$+CHR$135,0)
1620 PRINT':ENDPROC
1630 :
1640 DEF FNask(t$,a$,Y%)
1650 PROCcen(CHR$130+t$+"?"+CHR$131,Y%)
1660 LOCAL A%,B%:A%=FNin(a$)
1670 REPEAT B%=A%:VDUASCMID$(a$,B%),8
1680 A%=FNin(a$+cr$):UNTIL A%>LENa$
1690 =B%
1700 :
1710 DEF PROCread(D%,T%,S%,B%):ctrl?0=D
%
1720 ctrl!1=B%:ctrl?5=3:ctrl?6=&53
1730 ctrl?7=T%:ctrl?8=S%:ctrl?9=&21
1740 PROCdfs(ctrl,10):ENDPROC
1750 :
1760 DEF PROCwrite(D%,T%,S%,B%):ctrl?0=
D%
1770 ctrl!1=B%:ctrl?5=3:ctrl?6=&4B
1780 ctrl?7=T%:ctrl?8=S%:ctrl?9=&21
1790 PROCdfs(ctrl,10):ENDPROC
```



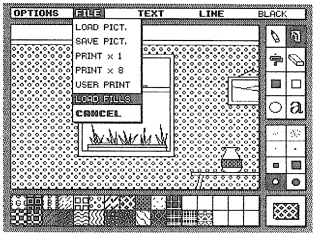

```
 1800 :
 1810 DEF PROCformat(D%
,T%,B%)
 1820 ctrl?0=D%:ctrl!1=
B%:ctrl?5=5:ctrl?6=&63
 1830 ctrl?7=T%:ctrl?8=
21:ctrl?9=&2A:ctrl?10=0
 1840 ctrl?11=16:PROCdf
s(ctrl,12):ENDPROC
 1850 :
 1860 DEF PROCsectid(T%,B%)
 1870 LOCAL S%,I%
 1880 FOR S%=0 TO sectors-1:I%=B%+S%*4
 1890 I%?0=T%:I%?1=0:I%?2=S%:I%?3=1
 1900 NEXT:ENDPROC
 1910 :
 1920 DEF PROCdfs(C%,E%):LOCAL A%,X%,Y%,
R%
 1930 A%=&7F:X%=C%:Y%=X%DIV256
 1940 REPEAT R%=R%+1:CALL osword
 1950 UNTIL C%?E%=0 OR R%=10
 1960 IF C%?E% IF error err=(C%?E%-6)DIV
2:*FX125
 1970 :ENDPROC
 1980 :
 1990 DEF PROCerr:*FX15
 2000 LOCAL Z%
 2010 PROCcen(CHR$129+e$(err),VPOS+4)
 2020 Z%=FNin(cr$):PROCcoff
 2030 err=0:ENDPROC
 2040 :
 2050 DEF PROCfserr:err=errs
 2060 e$(err)=FNreport
 2070 PROCerr:ENDPROC
 2080 :
 2090 DEF FNreport:LOCAL P%,a$
 2100 P%=((!&FD)AND&FFFF)+1
 2110 IF ?P%=0 =a$
 2120 REPEAT a$=a$+CHR$?P%
 2130 P%=P%+1:UNTIL ?P%=0:=a$
 2140 :
 2150 DEF PROCmake:PROChead(T$)
 2160 LOCAL drive40,drive80,Y%,Z%
 2170 Y%=VPOS+2
 2180 drive80=FNask("Which drive is 80 t
rack (0-3)","0123",Y%)-1
 2190 drive40=FNask("Which drive is 40 t
rack (0-3)","0123",Y%+2)-1
 2200 PROCclean(Y%,Y%+2)
 2210 PROCcen(CHR$130+"Insert disc in 40
 track drive",Y%)
 2220 Z%=FNin(cr$)
 2230 PROCdatachk(drive40)
 2240 PROCdotracks(drive40,0,0,FALSE)
 2250 PROCdotracks(drive40,5,9,drive40=d
rive80)
 2260 PROCdotracks(drive40,20,39,drive40
=drive80)
 2270 PROCcen(CHR$130+"Insert disc in 80
 track drive",Y%)
 2280 Z%=FNin(cr$)
 2290 PROCdotracks(drive80,5,9,FALSE)
 2300 PROCdotracks(drive80,20,39,FALSE)
 2310 ENDPROC
 2320 :
 2330 DEF PROCdotracks(D%,F%,T%,step%)
 2340 FOR track%=F% TO T%
 2350 PROCcen(CHR$134+"Track "+STR$track
%,Y%+2)
 2360 IF step% trak%=track%*2 ELSE trak%
=track%
 2370 PROCsectid(track%,buffer)
 2380 PROCformat(D%,trak%,buffer)
 2390 NEXT
 2400 IF F%=0 PROCdiskinit(D%,40)
 2410 PROCclean(Y%+2,Y%+4):ENDPROC
 2420 :
 2430 DEF PROCdatachk(D%)
 2440 LOCAL F%,Y%,Z%,error:Y%=VPOS+2
 2450 PROCclear(buffer)
 2460 PROCread(D%,0,1,buffer)
 2470 F%=buffer?5 DIV8
 2480 IF F%=0 OR buffer?5=&E5 ENDPROC
 2490 PROCcen(CHR$131+"Disc contains dat
a",Y%+2)
 2500 PROCcen(CHR$131+"Escape to abort",
Y%+4)
 2510 Z%=FNin(cr$):PROCclean(Y%+2,Y%+4)
 2520 ENDPROC
 2530 :
 2540 DEF PROCdiskinit(D%,T%)
 2550 PROCcen(CHR$129+"Initialising...",
Y%+4)
 2560 PROCclear(buffer)
 2570 $(buffer+8)="dummy  "
 2580 $(buffer+16)="dummy  "
 2590 buffer?15=ASC"$" OR &80
 2600 buffer?23=ASC"$" OR &80
 2610 PROCwrite(D%,0,0,buffer)
 2620 PROCclear(buffer)
 2630 buffer?5=16
 2640 buffer?6=(sectors*T%)DIV256
 2650 buffer?7=sectors*T%
 2660 buffer!12=&64006400
 2670 buffer!20=&0A002800
 2680 PROCwrite(D%,0,1,buffer)
 2690 ENDPROC
 2700 :
 2710 DEF PROCclear(B%):LOCAL L%
 2720 FORL%=0TO255STEP4:B%!L%=0:NEXT
 2730 ENDPROC
 2740 :
 2750 DEF PROCclean(Y%,Z%):LOCAL X%
 2760 FOR X%=Y% TO Z%:PRINTTAB(0,X%)SPC4
0;
 2770 NEXT:VDU31,0,Y%:ENDPROC
 2780 :
 2790 DEF FNagain:PROChead(T$)
 2800 =FNask("Another disc","YyNn",6)<3
```

Product : AMX Utilities
AMX Desk
Supplier : AMS
Green Lanes, Appleton,
Warrington, WA4 5NG.
Price : £14.95 (disc)
£24.95 (disc)

Product : Colour Art
Supplier : Watford Electronics
250, Lower High Street,
Watford,
Herts.
Price : £14.95 (disc)

The AMX Mouse grows up

The AMX Mouse has already captured the imagination of many Beeb owners. Geoff Bains test drives the latest rodent software from AMS and Watford Electronics.



With the release of AMX Utilities and AMX Desk the AMX mouse comes of age. These two packages enable the mouse fan to make his Beeb even more of a Macintosh.

AMX UTILITIES

The AMX Utilities package comprises additional utilities that complement the AMX Art package supplied with the mouse. It could be said that this package fills in all the gaps left in AMX Art, but that perhaps is a little harsh. The AMX Utilities package comes complete with an improved AMX Art program, and an identical copy of the old Design program, that completely replace your old software. In addition to these, there is a colour sketch program, a fill pattern designer, picture border stripper, a 'slide show' generator (could it be that AMS is an ardent BEEBUG watcher?) and the utilities suite itself.

Like the AMX Art package, the software is supplied on a 40 track disc with a program to convert it to 80 tracks. Unfortunately, the disc used has no write-protect notch, so this is not possible. However, a pair of scissors soon sorts out that problem.

There are two changes in the new AMX Art program. Firstly, a bug in the old program that affected long term use has been cleared up and, secondly, the option of loading up different fill patterns is provided.

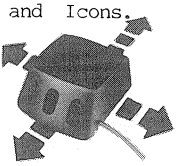
New fill patterns of your own design are created with the separate Design program. This is very much like the original icon design program and so you can leap into it without a glance at the manual. All good stuff. Although the old set patterns are good there is no substitute for your own ideas. Now you can, say, fill in an area of the screen with a frieze of your initials.

Another enhancement of AMX Art is the option to call up a printer dump routine in one of the many utility ROMs that you might have in your machine.

The slide show generator displays a series of specified pictures generated with AMX Art one after another, either automatically or at the press of a (mouse) button.

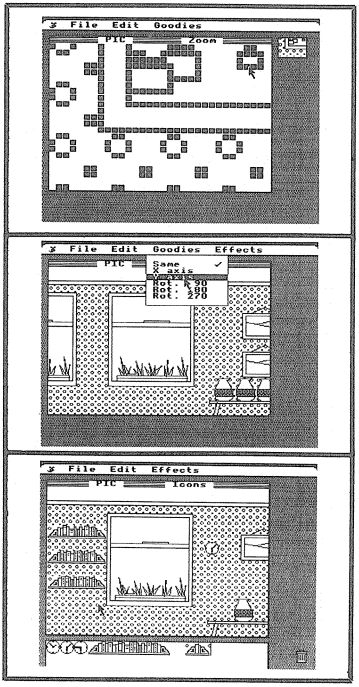
The major reason for buying AMX Utilities, however, has to be the utilities suite. Although these are used to operate on AMX Art pictures, the suite is self contained. It is not called from within AMX Art - a pity as this makes hopping between creating a picture and fiddling with it using the utilities much more cumbersome. It would have been nicer to have all the utilities on menu in AMX Art and overlay them from disc.

Calling up AMX Utilities presents you with a screen with three pull down menus at the top and a picture space in the centre. There are four utilities provided - Zoom, Copy, Curves, and Icons.



Zoom provides a facility that was obvious by its absence from AMX Art. Now you can enlarge a

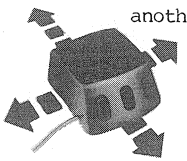
small section of your picture to full screen size and operate at pixel level. Locating the cursor over a single pixel space (now a character size) and pressing a mouse button will put a pixel in there, if there isn't one already, or wipe out the pixel already there. This allows you to put that final exact touch to your pictures that make them look that much more professional, especially in printer dump form where every slip shows up.



The Copy utility is a lovely addition. Any area of your picture can be copied onto another place in the same picture after being rotated through any number of right angles, or reflected in either axis. That's not all. The copy can either be drawn over whatever is in the destination position, inverted, or the space wiped clear first.

The Curves facility is another that was sadly missing from the original AMX Art. Whereas that program allowed only full circles to be drawn, this utility gives you arcs and ellipses, again with the invert, over, or wipe options. The inability to use this facility from the main AMX Art program reduces its usefulness very considerably.

The last utility in the suite is titled Icons. This uses one or more files of icons created with the Design program (components and music files are provided as starters) to build up pictures or to add to existing pictures. If your skill with the mouse is not quite up to scratch, a 'gridlock' system can be brought into effect to accurately position the icons, aligned with an invisible grid. Again, icons can be positioned over one another, inverted, or overwrite previous icons. Combined with the ease and naturalness of the mouse, this has to be the best method of producing technical diagrams available for the non-draughtsman.



AMX DESK

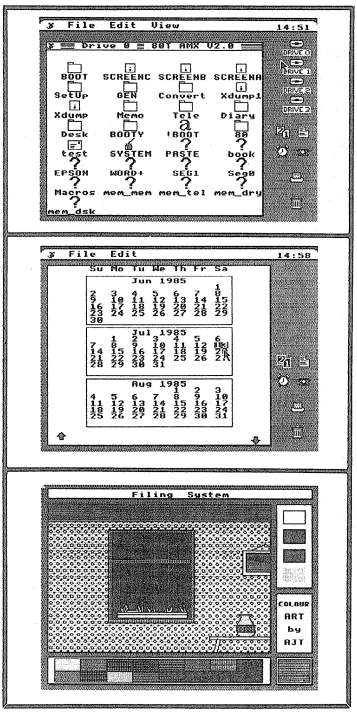
How Macintosh can a Beeb get? With AMX Desk the answer has to be very. AMX Desk gives your Beeb all the kinds of desk-top management features that Apple is so proud of pushing in its adverts. AMX Desk is really several different programs combined together in an easy-to-use, mouse-driven package.

First up (literally) is a clock and calendar. On boot-up you are asked the date and time (unfortunately you must type it in - you can't just press Return if you're in a hurry) and this is remembered until you next do a Ctrl-Break. Then the typical desk-top simulation is presented with icons for the drives, the diary/calendar, telephone number list, and memo pad. A calculator is also included but this is hidden away in one of the pull down menus and isn't always available. If you use a second processor then the calculator is available at all times. Either way, using the mouse to position the cursor over the buttons of an iconic calculator has to be the worst way of getting your sums right, so it's no great loss. Selecting a disc icon will give a catalogue of the disc complete with icons to describe the types of file there. Selecting an item directly from this catalogue, with the mouse, will CHAIN or *RUN it as appropriate.

The diary/calendar is a lovely piece of programming. It gives you a calendar of three months starting at the present. This can be scrolled back and forth across a two year period and any day selected. Now three days are displayed, and entries can be made from the keyboard. Any day with an entry is shown in inverse video in the main calendar. The whole lot is stored to disc, so your complete appointments can be filed away without ever putting pen to paper.

An alarm is also used that enables your Beeb to prompt you at any time of any day that an appointment is due. The usefulness of this is lessened by the fact that it will only operate if you are using AMX Desk at the time the alarm is due.

Memopad is effectively a mini word processor. Up to three pages of 24 lines of 32 characters can be filled with text and saved onto disc. Simple editing facilities are provided along with centring and justification - ideal for scribblings! More advanced facilities enable you to take sections of the memo and 'paste' them into another memo or even into a diary entry.



The third major part of AMX Desk is the telephone/ address 'book'. This is a simple database for brief details on all your contacts. The initial screen of this program is designed like the real thing with an alphabetic index down the side. Selecting any letter with the mouse displays the entries for that letter which can also be printed out for a more permanent record.

As a suite of useful everyday programs AMX Desk excels. However, it is not on to stretch it to the more arduous task of providing an icon-based environment from which to use your Beeb. That is where AMX departs from Mac. To give them credit, AMS and Elliot Software have done their best to establish the AMX system at the lowest of levels, but without re-writing the OS this is not that practical. If you are looking for a genuine Macintosh, you will have to spend the £2000 odd needed. If, however, you are happy with having to know a fair bit of the 'technical' operation of a computer but still want the easy front end, AMX is the answer.

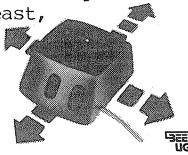
COLOUR ART

Watford's Colour Art is just the kind of program that it would have been nice to have seen in the AMX Utilities package. Using Colour Art, pictures created with AMX Art (and the Utilities for that matter) or any other similar package can be coloured in four colours. There are two clever points to this software. Firstly, the mode 4 AMX Art picture is, on loading, converted by the program into a mode 1 picture in two colours, whereupon it can be coloured in with the remaining two colours of mode 1.

The other clever trick is that, although only four colours in total are used, the package gives you a menu of sixteen shades, created with stipple patterns, to choose from. The four 'primary' colours can also be changed in the normal VDU19 type way and this data saved with the picture.

The Colour Art package, of course, uses the mouse to actuate all the options. Shades are selected by moving a (rather grotty) cursor over the shade menu boxes, pressing the Execute button, and then moving to the area that needs colouring. This area can be any part of the picture bounded with a solid line. Even an area already 'filled' with a monochrome AMX Art pattern can be re-filled in colour.

The intention is obviously to make the Colour Art package emulate (in colour) the original AMX Art. However, the operation is not as slick as the original and the presentation generally inferior. There are no beautifully designed icons on the screen in this work, only a flickering cross cursor. However, Colour Art is not a poor program and provides a highly novel and useful way in which to obtain high resolution colour pictures using the mouse. For that, at least, Watford Electronics must be congratulated.



BBC POP-UP CALCULATOR

With all the power of the micro at your finger tips, what better than an instantaneous pop-up calculator that can be called up when in the middle of using Wordwise or some other program. David Pilling explains.

Very often, either when using an application program or editing a program on your computer, the need arises to do some trivial calculations. Despite having a computer in front of you, the usual way of coping with this situation is to keep a calculator by the side of the machine. The reason is that the computer is tied up doing something else and you can't therefore use it. In recent times this problem has led to the availability of programs for personal computers which at the press of a key allow a set of tools like a notepad or telephone book to be used in the middle of another program.

So far such facilities are not generally available on home computers. It would seem therefore that their users must hang onto their old pocket calculators. All is not lost however, as the program presented here will turn a BBC micro into a fair approximation of a standard four function calculator, which can be used from within another program.

Suppose that you are in the middle of a program. Pressing the @ key will cause the calculator to appear in the top left hand corner of the screen, identifying itself with a # prompt. You can now go ahead and calculate whatever you like. Pressing Tab will return you to the original program. The calculator will clear a window of 15 characters, and your calculations should be kept within this window otherwise the program will not be able to clear it all.

The following keys are used to control the calculator;

+: for addition
-: for subtraction
*: for multiplication
/: for division
space bar: for clear

If you wish to use negative numbers (i.e 64*-20) then the underscore (_ key) should be used for the minus sign.

Return is the equivalent of the normal "equals" key. Numbers can be entered in the usual floating point notation using +, _ (underscore), E and . where appropriate. Delete will also work as usual algebraic notation is used. Please note that as with a normal calculator, the current operation (e.g. +,-,*,/) does not appear in the window.

The calculator should work with your own programs and we have also used it to good effect with Wordwise and Wordwise Plus for example, although we seem to have had difficulty with Wordwise Plus 'hanging', not only with the calculator but with some other software. There is no real explanation for this at the moment, but if you do have trouble, then try clearing your machine before running the program.

SETTING UP THE CALCULATOR

To get this useful facility into your computer, type in the program and then save it. When run, this program will assemble the machine code for the calculator and change the necessary vectors to point to the calculator routine. As soon as the program has been run, you can test the calculator, by pressing @. If there are any errors in the program, then before correcting them and re-running it, you must press Break (to reset the vectors) and then type OLD.

PROGRAM NOTES

The program works by enabling the 'character entering buffer' event and then intercepting the event vector (at &220). In fact, the event is used in an interesting way. The usual problem with events or interrupts, is that the code executed by them must only take a short amount of time if the interrupt system is not to be upset and in the worse case bring the machine to a grinding halt.

However, when actually in the event, the return address is kept 21 places down the stack. By storing this in the locations rl and rh (lines 1100 and 1130) and then replacing it with the start address of the calculator (lines 1110 and 1140), it is possible to arrange for the calculator to be re-entered on return from the event. To return to the original program, an indirect jump to the location in rl, rh is done (line 1250).

The calculator makes use of the floating point subroutines in the BBC Basic ROM. Since these are in different positions in BASIC I and BASIC II the program detects which version of Basic is in the computer (in line 100) and then uses either PROCB1 (line 2520) or PROCB2 (line 2590) to provide the appropriate addresses.

The position of the machine code in memory is determined by the value in line 110. For disc users &900 should work in most cases, but cassette users will have to put the code somewhere else. One possibility is to put it &200 bytes below screen memory. So if you intend to work in mode 2, then line 110 should be changed as follows:

```
110 enter=&2E00
```

The corresponding value for each mode can be obtained by the following algorithm:

```
MODE n
PRINT ~HIMEM-&200
```

It is not possible to save the assembled machine code for future use unless the program were to be extended (using more than two pages of memory) to reset the event vectors. This extra memory will often not be available.

Now with this calculator available at the press of a key, you can put all of your old adding machines away and use your BBC micro. Alternatively, you might like to use the principle of the program as the basis for a routine of your own.

```
  10 REM POPUP CALCULATOR
  20 REM VERSON  B0.3
  30 REM BY D.J.Pilling
  40  REM BEEBUG  AUG/SEPT85
  50 REM PROGRAM SUBJECT
  60 REM TO COPYRIGHT
  70 :
 100 IF?&8015=50 PROCB2 ELS
E PROCB1
 110 enter=&900
 120 event=?&220+256*?&221
 130 IFevent=enter END
 140 PROCSETUP
 150 PROCASSEMBLE
 160 ?&220=evs
 170 ?&221=evs DIV256
 180 *FX14,2
 190 PROCINFO
 200 END
 210 :
1000 DEFPROCASSEMBLE
1010 FORI%=0TO3STEP3
1020 P%=enter:[OPT I%
1030
1040 .evs PHP:CMP#2:BNE evx
1050 PHA:TYA:CMP#at:BEQ eva
1060 PLA
1070 .evx PLP:JMP event
1080 .eva PHA:TXA:PHA
1090 TSX:TXA:CLC:ADC#21:TAY
1100 LDA&100,Y:STA rl
1110 LDA#cs MOD256:STA &100
,Y
1120 INY
1130 LDA&100,Y:STA rh
1140 LDA#cs DIV256:STA &100
,Y
1150 PLA:TAX:PLA:TAY:PLA:PL
P
1160 JMP event
1170
1180 .exit
1190 LDX rom:STX &FE30
1200 LDA#31:JSR OSWRITE
1210 PLA:JSR OSWRITE
1220 PLA:JSR OSWRITE
1230 LDA#14:LDX#2:JSR OSBYT
E
1240 PLA:TAX:PLA:TAY:PLA:PL
P
1250 JMP(rl)
1260
1270 .cs
1280 PHP:PHA:TYA:PHA:TXA:PH
A
1290 LDA#117:JSR OSBYTE
1300 TXA:AND#64:BNE cs2
1310 LDA#134:JSR OSBYTE:BNE
cs3
1320 .cs2 LDX&364:LDY&365
1330 .cs3 TYA:PHA:TXA:PHA
1340 LDA#13:LDX#2:JSR OSBYTE
1350 LDA#252:LDY#255:LDX#0:
JSR OSBYTE
1360 STX rom
1370 LDA#187:LDY#255:LDX#0:
JSR OSBYTE
1380 STX &FE30
1390
1400 .start LDA#0:LDX#4
1410 .sl STA fp,X:DEX:BPL s
l
1420 LDA#eq:STA col
1430 JSR cur
1440
1450 .lp
1460 JSR getn
1470 JSR inte
1480 LDA co2:STA col
1490 BNE lp
1500
1510 .inte
1520 LDA#fp DIV256:STA&4C
1530 LDA#fp MOD256:STA&4B
1540 LDA col
1550 CMP#A:BNE i2
1560 JSR add:JMP i8
1570 .i2 CMP#S:BNE i3
1580 JSR sub:JMP i8
1590 .i3 CMP#M:BNE i4
1600 JSR mul:JMP i8
1610 .i4 CMP#D:BNE i8
1620 LDX#4
```

```
 1630 .dtp LDA&31,X:BNE dtx:
DEX:BPL dtp
 1640 PLA:PLA:JMP start
 1650 .dtx JSR div
 1660 .i8
 1670 JSR str
 1680 LDA#0:STA&15:TAY:DEY
 1690 JSR f2a
 1700 JSR pri
 1710 RTS
 1720
 1730 .pri
 1740 JSR cur
 1750 LDX&36:LDY#0
 1760 .pp LDA st,Y:JSR OSWRI
TE
 1770 INY:DEX:BNE pp
 1780 RTS
 1790
 1800 .cur
 1810 JSR crx
 1820 LDX#18:LDA#32
 1830 .crl JSR OSWRITE:DEX:B
NE crl
 1840 JSR crx
 1850 RTS
 1860
 1870 .crx
 1880 LDA#31:JSR OSWRITE
 1890 LDA#0:JSR OSWRITE
 1900 LDA#0:JSR OSWRITE
 1910 LDA#35:JSR OSWRITE
 1920 RTS
 1930
 1940 .getn
 1950 LDA#0:STA CX
 1960 .np JSR get
 1970 CMP#E:BEQ n2
 1980 CMP#poi:BEQ n2
 1990 CMP#mis:BNE all:LDA #S
:JMP n2
 2000 .all CMP#D:BEQ n3
 2010 CMP#M:BEQ n3
 2020 CMP#S:BEQ n3
 2030 CMP#A:BEQ n3
 2040 CMP#C:BEQ n6
 2050 CMP#tab:BEQ ne
 2060 CMP#del:BEQ n8
 2070 CMP#eq:BEQ n3
 2080 CMP#48:BCC np
 2090 CMP#58:BCS np
 2100 .n2 LDX CX:BNE n9
 2110 PHA:JSR cur:PLA:LDX CX
 2120 .n9 STA st,X:INC CX
 2130 .n7 JSR OSWRITE
 2140 BNE np
 2150 .n3 TAY
 2160 LDA CX:BNE n4
 2170 STY col:JMP getn
 2180 .n4 STA&36
 2190 TYA:STA co2
 2200 JSR a2f
 2210 CMP#&40:BNE nx
 2220 JSR i2f
 2230 .nx RTS
 2240 .n6 PLA:PLA:JMP start
 2250 .n8 LDX CX:BEQ np
 2260 DEC CX:JMP n7
 2270 .ne PLA:PLA:JMP exit
 2280
 2290 .get
 2300 .gp JSR OSRDCH:BCS ge
 2310 RTS
 2320 .ge CMP#27:BNE gp
 2330 LDA#126:JSR OSBYTE:BNE
gp
 2340 ]
 2350 fp=FNS(5):col=FNS(1)
 2360 co2=FNS(1):CX=FNS(1):e
nd=P%
 2370 rom=FNS(1)
 2380 NEXT
 2390 ENDPROC
 2400 :
 2410 DEFFNS(N%):P%=P%+N%:=P
%-N%
 2420 :
 2430 DEFPROCSETUP
 2440 OSWRITE=&FFEE:OSBYTE=&
FFF4
 2450 OSRDCH=&FFE0
 2460 rl=&70:rh=&71:st=&600
 2470 mis=95:poi=46:E=69
 2480 eq=13:C=32:M=42:D=47:A
=43:S=45
 2490 del=127:at=64:tab=9
 2500 ENDPROC
 2510 :
 2520 DEFPROCB1
 2530 add=&A50E:sub=&A50B
 2540 div=&A6B8:mul=&A661
 2550 i2f=&A2AF:str=&A37E
 2560 a2f=&AC5A:f2a=&9ED0
 2570 ENDPROC
 2580 :
 2590 DEFPROCB2
 2600 add=&A500:sub=&A4FD
 2610 div=&A6AD:mul=&A656
 2620 i2f=&A2BE:str=&A38D
 2630 a2f=&AC34:f2a=&9EDF
 2640 ENDPROC
 2650 :
 2660 DEFPROCINFO
 2670 PRINT''''"Popup Calcula
tor located at &";~enter;"."
 2680 PRINT'"Length ";end-en
ter;" bytes."''
 2690 ENDPROC
```

7←

40/80 UTILITY - PROGRAM NOTES

This well structured program is in Basic, but uses several of the OSWORD calls for various disc routines. The recently released 'Advanced Disk User Guide' (on special offer to BEEBUG members - see BEEBUGSOFT adverts) also contains a wealth of information.

The most useful routines used (which could prove useful in other applications) are as follows:

PROCinitialise sets up global variables, buffer areas and error messages.

PROCread reads and PROCwrite writes a specified sector using the buffer.

PROCsetid sets up sector identification (including logical track) for use by PROCformat.

PROCdfs is a general OSWORD call using preset parameter block.

PROCmake produces a dual-format a disc, PROCdotracks formats a sequence of tracks in single or double-step mode, and PROCformat formats a specified physical track.

PROCdatachk checks for the presence of data by reading catalogue (if possible).

PROCdiskinit initialises the catalogue including entries for two dummy files.

Creative Sound

Will Creative Sound do for the budding musician what Creative Graphics did for graphics enthusiasts? Ian Waugh, the author of our recent series on Making Music on the Beeb, has been looking with some interest at this new book from Acornsoft.

Creative Sound by David Ellis and Chris Jordan, published by Acornsoft, at £9.95 for the book only, £17.95 for combined book and cassette, and £19.95 for book and twin dual-format discs.

David Ellis is a computer musician well known to readers of other (music) magazines. Chris Jordan designed the Beeb's sound system (yes, he's responsible for the 14 ENVELOPE parameters) and the Acorn Music 500 (see BEEBUG Vol.4 No.1).

The first two chapters describe the development of musical instruments, synthesisers and computer music. It includes such topics as psycho-acoustics, FM synthesis and Fourier synthesis, which may mean little to the reader, especially as such options are not possible on the Beeb. This would be the place to introduce the rudiments of music but such is not the case here, much to the relief of musicians I'm sure, but if you're not a musician you may struggle in later parts of the book.

Chapters 3 and 5 contain programs which allow you to experiment with the SOUND and ENVELOPE commands but no in-depth explanations are given. It is assumed the best form of teaching is by experimentation. I often wished that a few more 'sample inputs' had been included - perhaps I'm just lazy.

Chapter 4 includes a Music Interpreter which uses a simple yet quite powerful MCL (Music Composition Language). Chapter 9 develops this into a full Music Compiler which translates MCL data into music. Eleven music files are given including several original compositions and Ligeti's 'Continuum' - a fairly modern classical piece of considerable complexity.

Chapter 7 deals with real time music (playing the QWERTY keys) and contains four monophonic synthesiser programs. The 'Echo Synth' program, in particular, produces fascinating sounds and patterns.

Chapters 8 and 9 tackle computer composition. The 'Waltzes' program is based upon an idea, attributed to Mozart, in which 176 bars of notes are selected, not quite at random, to produce one of about 50 different waltzes. Six other compositional programs devise ways of keeping the random number generator within certain (musical?) limits, and some allow the user to influence the choice of notes. 'Phasemusic' duplicates the sort of effects used by Steve Reich and Terry Riley. A series of notes is played on two channels, synchronised at first but a delay is introduced to the second channel so the music gradually drifts out of phase. This delay must be very small and the 50ms minimum duration of the SOUND's duration parameter is too long. 'Qsystem', produces a machine-code fix to reduce D to 10ms. It stays in the machine, too, so you can use it with other programs.

The programs fully exploit the strengths of BBC Basic. Colour is not used at all, possibly to keep long programs as short as possible, but the omission results in bland displays. No programming concessions are made to the novice, or even the average, programmer.

Last year, four books devoted to making music on the BBC micro were published. They may have taken a little wind out of the sails (and possibly sales) of Creative Sound as many of the ideas in this book have been implemented elsewhere, albeit in different forms. If your knowledge of the Beeb's sound system and the rudiments of music is a little shaky you may benefit by studying one of the other books which explains them in more detail.

As a musician with a particular interest in making music with computers, I can't fail but to recommend Creative Sound. I would also strongly suggest you buy the programs, too. Many are quite long and, I suspect, not easy to debug.

ACORNSOFT'S BASIC EDITOR

Acorn at last concedes that there's more to editing basic programs than just using the Copy key. Geoff Bains, in a critical frame of mind, assesses Acornsoft's full screen editor.

Product : Basic Editor
Supplier : Acornsoft
654 Newmarket Road,
Cambridge.
0223-214411
Price : £29.90

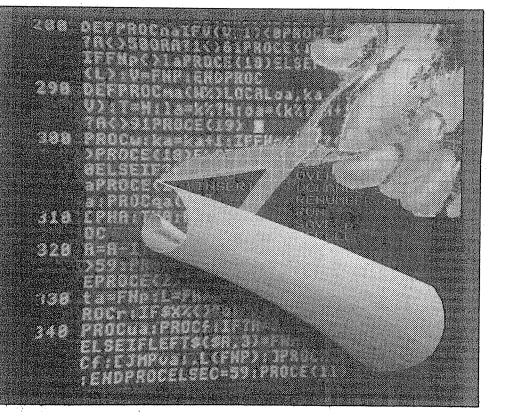
BBC Basic has a lot to be said for it. It's extensive, versatile, and fast. However, a major failing it has compared to the Basics of many other home computers is its editor. The simple screen copying technique that is used can be tedious in the extreme if a lot of editing is to be performed on a program.

How much better, I have mused to myself for about three years, if someone would come up with a ROM that would make the Beeb emulate the full screen editors in such machines as the Commodore 64 and MSX. Acornsoft has made a brave (if belated) attempt to answer my prayers.

The Basic Editor is a word processor for programs. In the package is a ROM chip, a key strip, and a forty page manual. The ROM is actually that, a ROM and not an EPROM which demonstrates that this version is here to stay.

The editor is entered with *BE and presents you with a View-like title screen displaying program size, bytes free, and so on. Just like View or Wordwise, you can now switch between this control screen and the program itself with the Escape key.

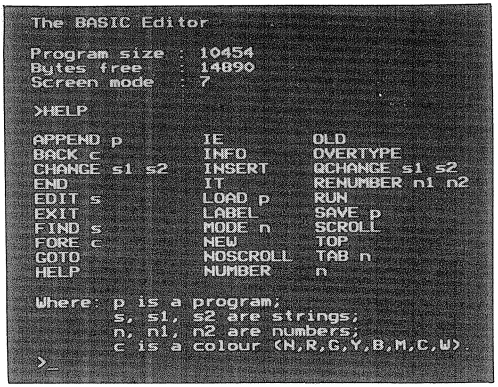
From the control screen some familiar sounding commands can be entered - LOAD, SAVE, NEW, OLD, and MODE - and a couple of new ones - HELP prints out a help screen with all the Basic Editor's commands, and INFO gives you the detailed low-down on the program in memory at the time.



A couple of nice touches have been added. Like View, no inverted commas are needed for filenames when loading and saving. The MODE command changes the display mode of the editing screen (the control screen stays in mode 7 all the time). In modes 3 and 6 this defaults to the striped blue background, so much favoured in 'The Computer Programme' and in other modes the plain blue background is used. Modes 2 and 5 are not allowed. The edit screen mode defaults, on entering the Basic Editor, to whatever mode your machine was in before leaving Basic.

From the edit screen, using the Basic Editor is just like using a word processor, all be it a somewhat warped one. The cursor keys move you freely around the screen and new program statements can be typed in at any position with an insert/overwrite option. This is switched with a function key. The function keys are made to work for their living with 28 options controlled using function keys and Shift or Ctrl. Many of these options are doubled by commands typed in full.

Where the Basic Editor shows its orientation to programs rather than text is in its handling of program lines. When editing a program you have no access to the line numbers. They are there mostly for show. Pressing Return anywhere in the program inserts a new program line which is automatically numbered for you. If there isn't enough space for a new line number, the whole program is renumbered automatically. That is fine, and it makes entering programs very easy (once you're used to it). However, things are not so simple when you want to go the other way.



To join program lines together a complicated manoeuvre of cursor movement and function key pressing is required to remove the offending line number.

Like a word processor, the Basic Editor has several commands that deal with blocks of program lines. The block between any two program lines, first marked with another function key press, can be copied or moved to another part of the program or simply deleted altogether. A nice feature is that a line can be tagged with another marker and the cursor moved to that line from another part of the program with a single key press.

The redundancy of line numbers is really shown up by the labelling system employed in the Basic Editor. When you're writing a program in the Basic Editor you can forget about line numbers and refer to previous or forthcoming lines in a program with a label. You can quite legitimately enter:

```
100 IF X>20 GOTO @BIGX
```

The @ symbol denotes a label name. At the point in the program pointed to you have to insert a REM statement of the form:

```
2000 REM @BIGX
```

When the program is next renumbered, or NUMBERed, the references to labels are replaced with line numbers as needed.

From the control screen several other advanced features are available. Most notable of these is the search and replace facility. There are many ROMs that include this (notably BEEBUGSOFT's Toolkit) but to have a search and replace included in the editor does make life easy.

The Basic Editor's search and replace is pretty sophisticated. There are four commands associated with this facility. EDIT <string> will present you with the first occurrence of the string in your program, in edit mode, and will take you on to the next occurrence with the press of a function key. FIND <string> will display all the program lines containing the string. CHANGE <string1> <string2> will replace all occurrences of string1 with string2 and lastly, QCHANGE <string1> <string2> will do the same but ask your approval in each case. What is sadly missing, however, is any kind of wildcard search facility. Only strings occurring in the program exactly as specified (taking account of letter case) will be 'found'.

There are several other commands available in the Basic Editor. These are mostly concerned with changing the default values of such things as the background and foreground colour, tab stops, and so on.

The Basic Editor operates well as a word processor for programs. If that is all you want then you will not be unhappy with Acornsoft's efforts. However, I must admit that I was a little disappointed. The Basic Editor is too much like a word processor and has become divorced from 'normal' Basic programming because of this. Although a program can be RUN from within the Basic Editor (in fact it returns you to Basic first), you have the feeling that you are not really in a programming language at all.

I would have preferred to have seen an editor that knitted more closely in with the existing Basic editing procedure, that used a normal LIST command to display a program, but included a real full screen editor. Such goodies as the search and replace could then be included as stand-alone utilities. As it is, the Basic Editor requires you to learn a whole new system of writing programs.

However, the Basic Editor is no mean feat in itself. Acornsoft may not have answered my own prayers in quite the way I wanted, but writing a Basic program on the Beeb is certainly easier now than it ever has been before.

Eprom Programmer Project

Geoff Bains, soldering iron in hand, describes the first stage in our constructional project to build an EPROM programmer. This month he introduces the subject and provides step-by-step instructions for constructing the hardware.

One of the Beeb's most versatile features is its ability to hold several system programs in its memory map, in ROM, at the same time, and call upon the services of any of these whenever needed. Many software houses have made good use of this system of sideways ROMs. However, many users will have requirements for which a commercial ROM is not available. Nor can they easily imitate the professionals and produce their own.

That is, until now. Starting this month, BEEBUG is presenting all you need to know about creating your own ROM software for your own particular purposes.

In the next few months we will be presenting software to produce the code to handle all the Beeb's complex ROM protocols from your own programs (both Basic and machine code), and we will be giving you examples of sideways software suitable for using in either sideways RAM or for blowing onto EPROMs. This month, however, we are starting the whole ball rolling with a DIY project to build your own EPROM programmer.

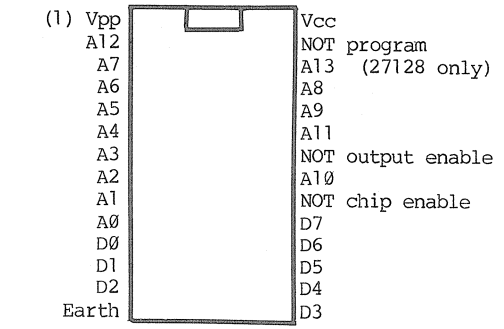
EPROMs are Erasable, Programmable, Read Only Memory chips. That is, like ROMs they can only be read from, and not written to, by your computer, but unlike ROMs they can, under special conditions, be both programmed and erased. The erasure of EPROMs is carried out with ultra violet light. Special units are available to do this to several EPROMs at once and, because of the inherent dangers of strong sources of ultraviolet light, it is recommended that you purchase a professionally made unit to do this. If you are totally sure of the validity of the code that you are to put into an EPROM then a new EPROM is suitable as these are purchased blank. However, if the code is to be changed at a later date then an eraser is a worthwhile investment.

The programming of EPROMs is another matter altogether. Commercial units are, of course, available for this task too but they can cost anything from about £50 to several hundred pounds for the really sophisticated models. The BEEBUG DIY EPROM Programmer will only cost you around £25 but offers all the facilities needed to program 8K or 16K EPROMs.

You can build the BEEBUG EPROM Programmer with one of two attitudes of mind. Either you can ignore all the technical details and just go ahead and put together the bits as will be described in a while. Alternatively, for those of you who want to know the technical details, this is how the whole device works.

THE TECHNICAL DETAILS

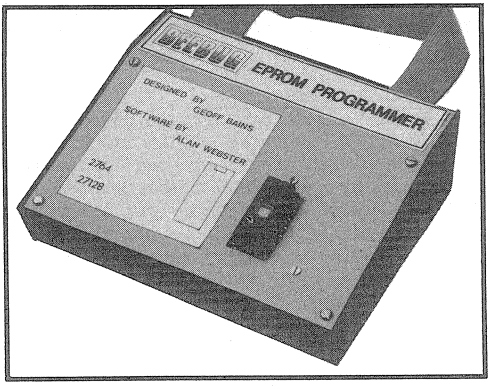
EPROMs, both 8K (2764) and 16K (27128) have the pin out as show in the diagram. The only difference between the two sizes is that the top address line, A13, is not included on the 8K EPROM.



To read an EPROM, the address lines (A0 to A13) are set up with the 14 bit value corresponding to the byte in the EPROM that you want to read, and the data is simply read off the eight data lines (D0 to D7). The 'output enable' and 'chip enable' lines are used to switch off, respectively, the data outputs and the whole chip.

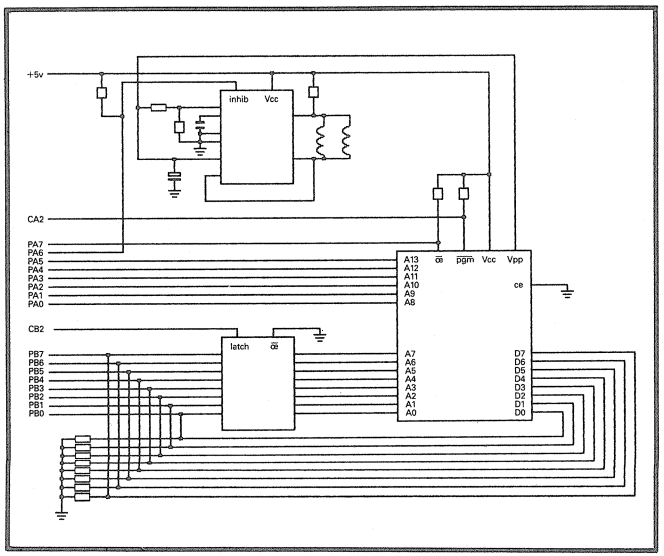
Programming an EPROM is quite a complex task. The address of the byte that is to be programmed is put onto the address lines, the data to be programmed is put onto the data lines, the Vpp line is taken to 21 volts and the NOT program line is taken low for 50 ms.

The BEEBUG EPROM Programmer will cater for both these tasks - reading EPROMs to study them and programming your own EPROMs.



THE CIRCUIT DIAGRAM

Using both the User Port and the Printer Port we have sixteen data lines available to control the programmer. However, we need twenty two - eight for the data lines and fourteen for the address lines. To get around this problem the 74374 octal latch chip is used. When the latch line is taken high this chip will latch the data at its input, and output it from its eight output pins. Using this chip the eight low bits of the address can be latched (using the User Port CB2 line) and then the eight User Port data lines are free for passing the EPROM's data. The six high address bits are taken care of by the Printer Port.



So that the data coming from the EPROM does not interfere with the low eight address bits before they are latched by the 74374, we need to be able to switch off the EPROM data output. This is taken care of by connecting the NOT output enable line of the EPROM to bit 7 of the Printer Port.

The 5 volt supply for the circuit is taken from the User Port. However, there is no 21 volt supply in the computer for EPROM programming. This is derived from the 5 volt line with the TL497A chip. This is a very useful device that can produce any voltage from about 6 to 30 volts from a 5 volts supply. It is also provided with an inhibit line to switch the output between the supply and output voltages. This we have connected to bit 6 of the Printer Port so that the computer can switch the programmer between read and write operations (5 and 21 volts).

Lastly the NOT program input to the EPROM is connected to the CA2 control line of the Printer Port. This has to be pulsed low for 50 ms to program a byte in the EPROM.

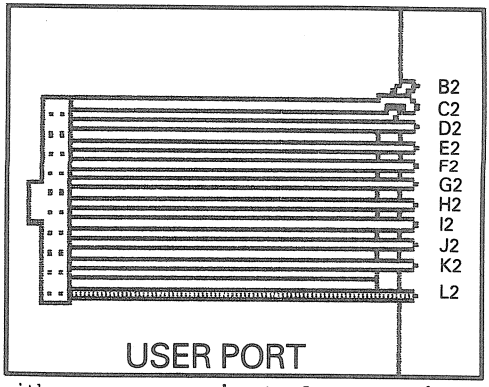
The eight resistors connected between the eight data lines and earth are to keep the data lines at a low level when no EPROM is connected. This means that the software (comming next month) can distinguish between a blank EPROM in the programmer (which contains

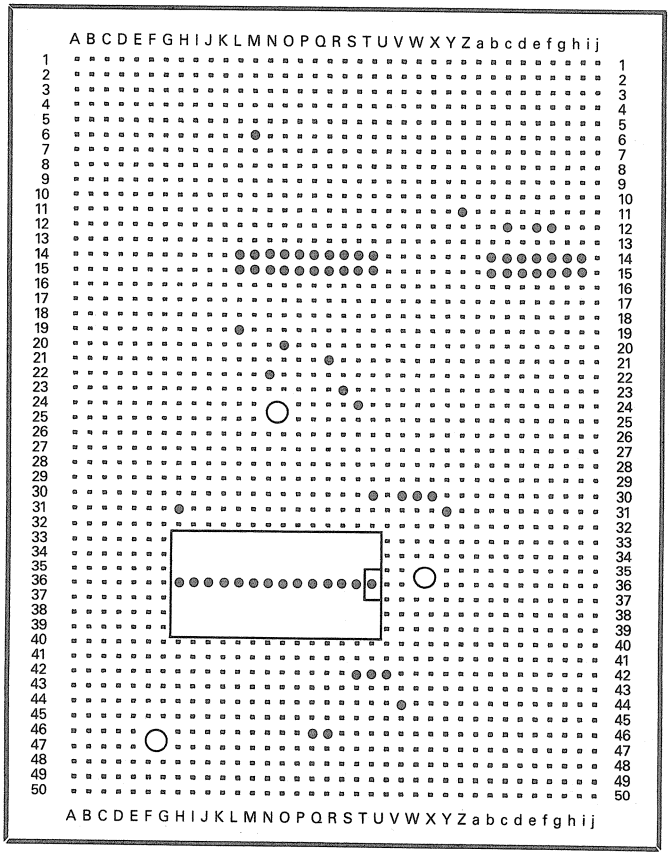
all bytes set to &FF) and no EPROM present at all (all bytes are held by the resistors at &00).

BUILDING THE PROGRAMMER

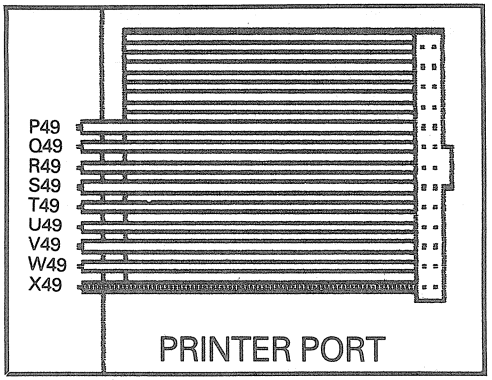
There is just one circuit board in the BEEBUG EPROM Programmer. This is built on stripboard so that it can be easily attempted by any BEEBUG member with just a little previous experience of soldering. At the end of this article you will see a list of the parts needed for the project (all bits are available from Maplin Electronic Supplies - tel. 0702-554155, or see the catalogues in W.H.Smith - or from other components stockists). You will also see two diagrams of the circuit board.

The first diagram, on this page, shows the assembled board from the side with the copper strips. This shows the cuts in the copper tracks that must be made first. Use either a proper 'spot face cutter' for this or a drill bit (about 0.3 inch). The second diagram, opposite, shows the component side of the board and the positions of all the parts. The success of your Programmer depends upon getting the right components into the right holes in the stripboard, so it is worthwhile spending some time on this. You may find it easiest to draw the positions of the track cuts and components onto the stripboard itself, making use of the co-ordinate system in the diagrams, before you actually undertake any irreversible actions like wielding the soldering iron or the spot face cutter. When you are satisfied that you have the layout right, make all the track cuts before you solder anything in place.





Solder the chip sockets in first. The socket for the EPROM itself is soldered onto the 'wrong' side of the board so that it will stick out from the copper side through the case, with the rest of the board flat against the case top. This is quite a tricky operation and should be done first. Position the socket correctly with the board, copper face up, on a table. Now carefully apply



the soldering iron and solder to each pin and its corresponding copper track, as they stand proud of the board.

The rest of the components should now be soldered into position. With the majority of them you need not worry which way round they are to go. However, the 47uF capacitor must be inserted the right way around (positive end towards the edge of the board, as shown) and the two chips must, of course, be plugged into their sockets correctly (see the notches at one end of the chips in the diagram).

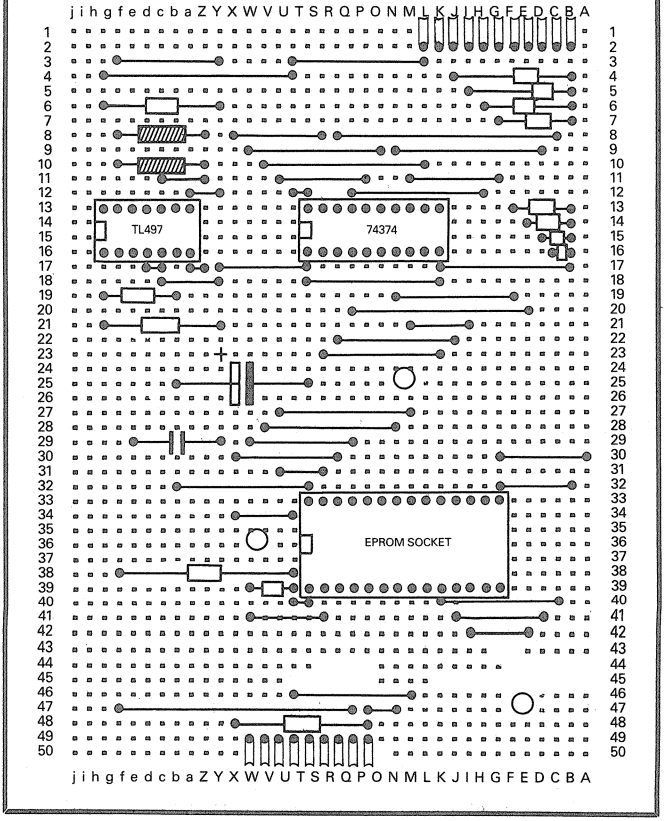
When you have finished the board and checked it thoroughly for any mistakes (check especially for solder 'bridges' between tracks), you can make up the cables. With the red stripe on the 20 way cable to the right, clamp the IDC socket around the cable with the locating lug uppermost. Repeat this procedure for the 26 way connector. Now solder eleven of the cores from the 20 way cable and nine from the 26 way cable as shown.

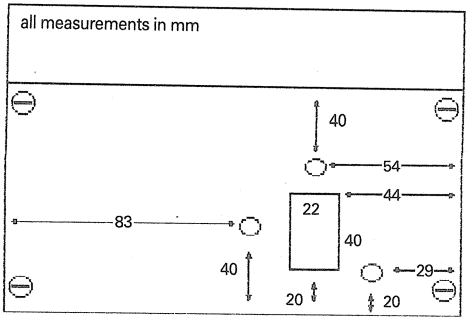
Follow the diagrams on these pages carefully. A wrong connection at this stage will mean that your EPROM Programmer will not operate correctly.

THE CASE

Now that the board is finished you can put the whole device into a suitable case. The choice of case is largely up to you. However, the case suggested in the parts list is a pleasing and practical solution. You will need to make three holes (about 0.25 inch) in the circuit board (where marked) and some cutouts in the lid of the case. For the box suggested the cutouts should be as shown in the diagram. Now take a piece of cardboard and cut it to the size of the lid and make similar cutouts in that.

Next the board should be bolted to the underside of the lid with the cardboard insulating the board from the metal and with the EPROM socket poking through the





rectangular hole. Make sure that you use a couple of the nylon washers on each screw on each side of the circuit board.

Finally the lid should be screwed back onto the case with the two ribbon cables running out of the back, clamped between the lid and back. Now your BEEBUG EPROM Programmer is ready for action. In the next issue we will be listing a sophisticated program to drive the programmer, to read, verify and program EPROMs.

Components List for BEEBUG EPROM Programmer

No.	Item	Maplin code	No.	Item	Maplin code
1	27374 octal latch chip	UB72P	1	reel 18 swg tinned wire	BL12N
1	TL497A DC to DC converter	YY78K	1m	20 way IDC ribbon cable	XR74R
1	1 ohm, Ø.4W resistor	M1R	1m	26 way IDC ribbon cable	XR75S
8	1 Kohm, Ø.4W resistor	M1K	1	20 way DIL IDC socket	FG84F
1	1.2 Kohm, Ø.4W resistor	M1K2	1	26 way DIL IDC socket	FG85G
3	4.7 Kohm, Ø.4W resistor	M4K7	1	50 hole x 36 strip stripboard	FLØ9K
1	20 Kohm, Ø.4W resistor	M20K	1	plastic case	LH66W
1	330 pF ceramic capacitor	WX62S	1	pack M4 x 12mm screws	BF49D
1	47 uF, 63v axial capacitor	FB39N	1	pack M4 nuts	BF57M
2	10 mH open wound choke	HX19V	1	pack nylon washers	BF83E
1	14 pin DIL socket	BL18U	1	spot face cutter	FL25C
1	20 pin DIL socket	HQ77J	1	reel solder	FR21X
1	28 pin DIL socket	BL21X			

A 28 pin ZIF socket can be purchased instead of the 28 pin DIL socket.

The resistors are identified by a code of colour bands. See the Maplin catalogue for an explanation of this.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS

MORE MEMORY - Douglas Siviter.

An extra 2K of memory can be gained by reducing the number of lines displayed in modes 4 and 5 to 25 (a usual number for other computers anyway). The following short program does this. HIMEM can then be increased by 2240 or the area used for a machine code program. The program works for modes Ø to 2 as well but in these cases you stand to gain 4480 bytes. Screen scrolling is slowed a little by this program.

```
10 MODE 4                :REM USE WHICHEVER MODE YOU LIKE
20 VDU23;6,25;0;0;0;     :REM SET 25 ROWS
30 VDU23;7,31;0;0;0;     :REM REPOSITION SMALL SCREEN
40 VDU28,0,24,19,0       :REM TEXT WINDOW
50 VDU29,0;224;          :REM GRAPHICS ORIGIN
```

FINDING FREE MEMORY - Richard Sterry

The first free location in RAM is given by 256*?3+?2 and the last free RAM address by 256*?5+?4. So free RAM is given by 256*(?5-?3)+?4-?2+1. This accounts for variable space, Basic stack, and screen memory and works with Basic I, II, and HiBasic.

1st course

Drawing simple bar-charts

Graphs provide much more striking displays than mere columns of figures. Chris Wragg describes a useful procedure for converting your data into colourful bar-charts, and all in mode 7.

Mode 7 is such a versatile mode that unless you need extra columns or really sophisticated graphics it is the obvious choice for many general purpose programs. As the example shows, some applications of graphics can work as well in mode 7 as in any of the so-called graphics modes.

The procedure listed here enables any suitable numeric data to be displayed in the form of a colourful bar-chart in mode 7. For example, I use this procedure in a program which analyses the costs of running a car and displays graphically my petrol consumption over the last 16 fill-ups.

DRAWING THE BAR-CHART

The complete graphical display is generated entirely by the procedure called PROCgraph in lines 1100 to 1350. This is written so that it can readily cope with a variety of scales and up to 30 vertical bars or columns. The parameters required for the procedure are:

title$	A title for the display.
low	Lowest value on the scale.
high	Highest value on the scale.
n%	Number of bars or columns.
colour%	Colour of the bar-chart.

Colour is a number in the range 1 to 7 as listed on page 223 of the User Guide.

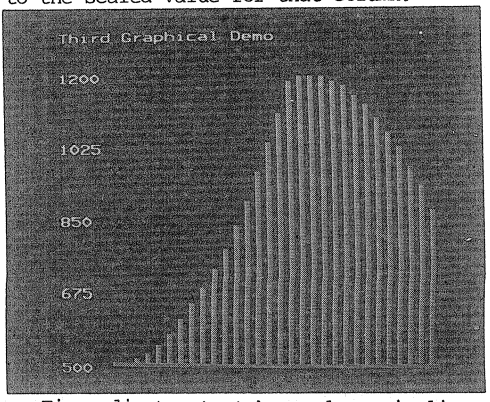
Since this is a mode 7 display, all the graphics are made up from suitable mode 7 graphics characters. The procedure displays first the title, and then calculates a scale factor to convert the data values to a screen value in the range 0 to 20 (the maximum height of a column in graphics characters). From the highest and lowest values, a vertical scale is produced at the left of the screen. For example, the call:

```
PROCgraph("1985",0,50,24,4)
```

would produce a bar-chart with 24 columns in blue, with a vertical scale running from 0 to 50, and with the title "1985".

To display graphics characters, a line of graphics colour codes is run vertically up the screen in column 4. Any graphics characters displayed to the right of these will be in the colour specified (see page 487 of the User Guide for more details of teletext colour codes).

After these preliminaries, the main display is produced by the loop commencing in line 1240. Each pass through this loop draws one column of the bar chart. The bottom of every bar is formed from graphics character 163 (see User Guide page 489). This produces the horizontal line across the bottom of the graph. Then a vertical column is drawn using graphics character 181 with a height corresponding to the scaled value for that column.



Fine adjustment takes place in lines 1300 to 1320. The main graphics character used, code 181, is a vertical line 3 pixels high (and 1 pixel wide). Graphics

characters 176 and 180 provide vertical lines 1 and 2 pixels high respectively, and these are used to top off each bar as appropriate and produce a more accurate result. To see the difference, just delete these lines from the procedure and re-run. This allows scaled values (in the range 0 to 20) to be represented to the nearest 0.3, rather than the nearest integer only.

Finally, the procedure places a vertical line of 'green text' characters (code 130) just to the right of the bar-chart. Any text then placed in this area is displayed in this colour (which you can change if you wish, or omit altogether).

THE DEMONSTRATION PROGRAM

The use of the procedure is demonstrated by the complete program listed here. In turn, this reads one of four sets of data into the array Item(), and then displays the data in bar-chart form on the screen. Each data set starts with the number of items. Pressing any key will immediately move the display on to the next bar-chart.

APPLICATION

The procedure, as already described, is quite flexible, and only requires that the data to be displayed be present in the array Item() before the procedure is called. There are many applications where a simple bar-chart is more effective than rows of figures, and the illustration should convince you that mode 7 can be a good choice for colourful and striking displays when used in the right way.

```
  10 REM Program CHART7
  20 REM Version B1.1
  30 REM Author Chris Wragg
  40 REM BEEBUG Aug/Sept 1985
  50 REM Program subject to copyright
  60 :
 100 MODE 7
 110 DIM Item(30)
 120 N%=FNread
 130 PROCgraph("First Display",10,15,N%
,5)
 140 PRINTTAB(20,4)"You can use this";T
AB(20,5)"space for text.";
 150 PRINTTAB(20,7)"If you are writing"
;TAB(20,8)"very much, use VDU28";TAB(20,
9)"to redefine the text";TAB(20,10)"wind
ow."
 160 PRINTTAB(20,16)"Press any key.";:A
=INKEY1000
 170 N%=FNread
 180 PROCgraph("Graph 2",0,100,N%,1)
 190 A=INKEY1000
 200 N%=FNread
 210 PROCgraph("Third Graphical Demo",5
00,1200,N%,2)
 220 A=INKEY1000
 230 N%=FNread
 240 PROCgraph("Fourth Demonstration",0
.5,3.5,N%,4)
 250 END
 260 :
 270 DATA 12,13,12.8,13.1,13.4,13.4,13.
7,14.0,14.1,14.0,14.3,14.7,14.9
 280 DATA 15,5,12,24,25,28,37,49,63,87,
97,85,77,61,45,13
 290 DATA 30,500,505,515,530,550,575,60
5,640,680,725,775,830,890,955,1025,1100,
1180,1190,1195,1190,1180,1165,1145,1120,
1090,1055,1015,970,920,865
 300 DATA 30,3.5,3.35,3.2,3.05,2.9,2.76
,2.62,2.48,2.35,2.22,2.10,1.98,1.87,1.76
,1.66,1.56,1.47,1.38,1.30,1.22,1.15,1.08
,1.02,.96,.91,.86,.82,.78,.75,.72,.7
 310 :
1000 DEF FNread
1010 LOCAL i%,n%:READ n%
1020 FOR i%=1 TO n%:READ Item(i%):NEXT
1030 =n%
1040 :
1100 DEF PROCgraph(title$,low,high,n%,c
olour%)
1110 CLS:PRINTTAB(0,0)title$;
1120 LOCAL r,scale,X%,Y%,H,H1
1130 REM determine vertical scale
1140 r=high-low:scale=20/r
1150 REM display scale values
1160 PRINTTAB(0,23);low
1170 PRINTTAB(0,18);low+(0.25*r)
1180 PRINTTAB(0,13);low+(0.5*r)
1190 PRINTTAB(0,8);low+(0.75*r)
1200 PRINTTAB(0,3);high
1210 REM set colour for graph
1220 FOR Y%=1 TO 23:PRINTTAB(4,Y%);CHR$
(145+colour%);:NEXT
1230 REM draw bar chart
1240 FOR X%=5 TO n%+4
1250 PRINTTAB(X%,23);CHR$163;
1260 H=(Item(X%-4)-low)*scale
1270 REM draw column
1280 IF H<1 GOTO 1300
1290 FOR Y%=1 TO H:PRINTTAB(X%,23-Y%);C
HR$181;:NEXT
1300 H1=H-INT(H)
1310 IF H1>.3 AND H1<.6 PRINTTAB(X%,22-
INT(H));CHR$176;
1320 IF H1>=.6 PRINTTAB(X%,22-INT(H));C
HR$180;
1330 NEXT X%
1340 FOR X%=0 TO 23:PRINTTAB(n%+5,X%);C
HR$130;:NEXT
1350 ENDPROC
```

Adding New OS Commands

T.A. Lucas shows just how easy it is to add new commands and functions to the Beeb's operating system.

THE BBC SYSTEM.

A very useful feature of the BBC microcomputer's operating system is the * command. If a * command such as *LOAD is issued from the keyboard, or from within a program, the text after the * character is offered first to the operating system. If the operating system (O.S.) recognises the command, it will be executed without being passed to any other part of the system. If the text after the * character is not recognised by the operating system, for example *XYZY, the text is offered to the paged ROMs. If none of these recognise the command, the operating system makes a call to the Operating System Filing System Control (OSFSC) vector with the accumulator set to 3. The filing system will then attempt to *RUN the unrecognised command. If the filing system cannot *RUN the file requested, the error "Bad command" is issued. Thus a * command may call an O.S. or DFS routine, another (sideways) ROM, or a disc-based routine.

INTERFACING THE NEW COMMAND.

For disc users this is a very useful and powerful feature. It enables machine code routines to be called from disc into memory only when they are needed. This means that the machine code can be already assembled, rather than needing assembly at run-time, and as a result the execution address does not need to be known by the calling program. Thus the routine can be changed or relocated without any need to change the call address in the main program.

However, it may be desirable to have a machine code routine which is used frequently during execution of a program. The delay incurred in fetching the routine from disc every time is then unacceptable. A method is needed by which the program can be loaded from disc the first time it is called, and then held in memory so that it will be recognised the next time the command is issued. Thus the routine would not need to be loaded from disc again. The program should conform to the following criteria:

1. The command may be in upper or lower case.
2. The command may be made as a * command, using the Basic OSCLI function available in Basic II, or using the operating system entry vector at OSCLI.
3. A parameter list may be supplied after the call, provided it is separated from the command by a space.
4. The routine may reside in either the I/O processor or a second processor and be called from the second processor.

The last feature is extremely useful as there is no other method of calling a machine code routine directly in the I/O processor from the second processor.

If the code resides in the second processor, care must be taken to ensure it is Tube compatible.

Since the routine is intended to be called from disc, one useful place to locate the code is at the top of the disc filing system area. Up to two pages are available in most circumstances from locations &1700 to &18FF. Alternatively, when a second processor is fitted, there is a large area of memory, from &1F00 to &2FFF in mode 0, which may be used for one's own routines. It is not recommended that all of this is used because some room must be left for certain disc functions, such as *COMPACT and *COPY, to operate.

RUNNING THE EXAMPLE COMMAND.

As an example of this technique, the program given should be typed in and saved. When you run the program the routine is assembled and can then be saved on disc by copying the details of the *SAVE command produced by the program. Once saved in this way, the code can be called by any of the following methods:

1. *DEMO <Parameters>
2. OSCLI("DEMO <Parameters>")
   - only available in Basic II.
3. DIM P% 255:$P%="DEMO <Parameters>"
   :X%=P% MOD 256:Y%=P% DIV 256:CALL
   &FFF7 - using OSCLI vector.

The first time the command is executed, the program will be loaded from disc and start execution at label "ExAddr". This section of code will reset the value of OSHWM if necessary to ensure that the program is not overwritten by calls such as *COMPACT and *COPY. The program then modifies the command line interpreter vector at locations &208 and &209. These are made to point to the new routine, having saved the previous contents of these locations. This is necessary to ensure that any * command not recognised by our new routine is passed back to the operating system or on to the next routine. If the command is recognised, the parameter list is interpreted. In the example given, the string following the * command is printed vertically on the screen (e.g. the command *DEMO BEEBUG will print 'BEEBUG' down the screen).

When the command is issued again, the response is instantaneous because the command is now recognised as being in memory, and the routine does not have to be loaded from disc.

ADDING YOUR OWN COMMANDS

To add your own * commands into a program, you will need to delete the machine code following the label '.ThisCmd' in line 2000 up to line 8999. You should then put your own machine code between these lines and alter the name of the * command at line 100. A quick example will show you how to make the micro beep upon receiving the command *BEEP.

1. Delete lines 2000 to 8999 by typing DELETE 2000,8999
2. Change line 100 to:
   100 Name$="BEEP"
3. Insert your machine code:
   2000 .ThisCmd
   2010 LDA #7
   2020 JSR &FFEE
   2030 RTS

Then RUN the program and the command *BEEP will now produce a noise. By saving the assembled code on disc (details are given by the program itself), the *BEEP command could be used in any other program, and will load and execute the assembled code from disc as required.

```
10 REM PROGRAM NEW OS COMMANDS
20 REM VERSION B0.4
30 REM AUTHOR  T.A. LUCAS.
40 REM BEEBUG  AUG/SEPT '85
50 REM PROGRAM SUBJECT TO COPYRIGHT.
60 :
100 Name$="DEMO":REM Name for command
and object file.
110 OSBYTE=&FFF4:OSWRCH=&FFEE:OSNEWL=&
FFE7
120 ZPage=&F2:CLIV=&208
130 Start=&1900-&200:REM Allow 2 pages
 below DFS.
140 PROCAssemble
150 @%=&90A
160 PRINT" *SAVE "Name$;" "~Start;" "~
P%;" FF"~ExAddr;" FF"~Start
170 END
180 :
1000 DEFPROCAssemble
1010 Cmd=Start:REM Location of name of
command.
1020 $Cmd=Name$
1030 FOR Pass%=0TO3STEP3
1040 P%=Start+LEN(Name$)
1050 [OPT Pass%
1060 .NewCLI STX ZPage:STY ZPage+1\ Sav
e pointers to command.
1070 LDX #0:LDY #0
1080 JSR FetchChar\ Skip any spaces.
1090 CMP #&2A:BNE CompCmd
1100 INY:JSR FetchChar\ Skip "*" charac
ter.
1110 .CompCmd LDA (ZPage),Y:AND #95
1120 CMP Cmd,X:BNE NotThisCmd
1130 INY:INX:CPX #LEN(Name$)
1140 BCC CompCmd\ Test full length of n
ame.
1150 LDA (ZPage),Y:CMP #13:BEQ ThisCmd\
 At end of command.
1160 INY:CMP #32:BEQ ThisCmd
1170 .NotThisCmd LDX ZPage:LDY ZPage+1
1180 JMP (OldCLIV)\ Try next routine.
1190 .FetchChar LDA (ZPage),Y:CMP #32
1200 BNE ExitFetchChar
1210 INY:BNE FetchChar\ Skip spaces.
1220 .ExitFetchChar RTS
1230 .OldCLIV NOP:NOP\ Store for old ve
ctor
1240 .ExAddr LDA CLIV:STA OldCLIV
1250 LDA CLIV+1:STA OldCLIV+1\ Save old
 vector.
1260 LDA #NewCLI MOD256:STA CLIV
1270 LDA #NewCLI DIV256:STA CLIV+1\ Set
up new vector.
1280 TYA:PHA\ Save command offset.
1290 LDA #180:LDX #0:LDY #255:JSR OSBYT
E\ Fetch OSHWM.
1300 CPX #(ExAddr DIV256)+1:BCS OSHWMSe
t\ Test if routine protected.
1310 LDX #(ExAddr DIV256)+1:LDY #0:JSR
OSBYTE\ Set new value for OSHWM.
```

```
 1320 .OSHWMSet PLA:TAY\ Restore pointer
.
 1330 :
 2000 .ThisCmd TYA:PHA\ Save pointer to
command parameters.
 2010 LDA #134:JSR OSBYTE\ Fetch cursor
position.
 2020 PLA:TAY\ Restore pointer.
 2030 .Display LDA (ZPage),Y:CMP #13:BEQ
 ExitCmd
 2040 JSR OSWRCH\ Display character.
 2050 JSR OSNEWL
 2060 TXA:BEQ TabSet\ Test for tabbing n
eeded.
 2070 PHA\ Save tab position.
 2080 LDA #9\ Cursor forward one charact
er.
 2090 .SetTab JSR OSWRCH
 2100 DEX:BNE SetTab
 2110 PLA:TAX\ Restore X register.
 2120 .TabSet INY:BNE Display
 2130 .ExitCmd RTS
 9000 ]
 9010 NEXT
 9020 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS

DECIMALS TO FRACTIONS - Jonathon Wilson

The short program below takes as input a decimal number (e.g. 1.125), and outputs the lowest common denominator fraction (9/8 for the example above). Note that values of zero will produce the response 'Silly'.

```
10 INPUT"Decimal "D:T%=D*1E7:B%=1E7
20 REPEAT READC%:IF (T%MODC%=0) AND (B%MODC%=0) T%=T%/C%:B%=B%/C%:RESTORE
30 UNTIL C%=97:IF T%=0 PRINT "Silly" ELSE PRINT "Fraction ";T%;"/";B%
40 DATA 2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,97
```

INVISIBLE BOOTS - Paul Baron

If you don't like your !BOOT files to write PAGE=&1900 and CHAIN"MENU" all over the nice clean screen, here is one method of preventing it. When *BUILDing the file, create the first line as XXXXX. Now use a disc sector editor such as in Disc Doctor, Watford DFS or BEEBUG Vol.2 No.3, to alter the Xs to the hex values:

```
1C 01 07 01 07
```

These are the VDU codes to create a zero sized text window. As it is the codes themselves that are used, nothing appears on the screen. The first thing that the menu program (or whatever) should do is to change the mode or text window, otherwise all screen output will remain invisible.

SWITCHING DISC DRIVES - Alan Webster

This function key definition will set up one of the function keys to switch between disc drives. Pressing the key chosen (according to the value substituted for 'n') will switch over to the next drive until drive 3 is reached when a further key-press will bring you back to drive 0 again. Keeping the Tab key down at the same time will give a catalogue of that drive. This routine will only work with Basic II because of its use of the OSCLI command.

```
10 *KEYnDIM X% 6,S% 10:X%?0=0:X%!1=S%:X%!5=1:A%=6:Y%=X%DIV256:CA.&FFD1:D%=(3A.((S%?1
)+1)):OSCLI("DR."+STR$(D%)):P.CHR$12"Drive ";D%:IFINKEY(-97)OSCLI(".")|M
```

UNDERLINING CENTRED TEXT - Ian Tressman

Wordwise just prints spaces before a piece of text, to centre it. If your printer's underline feature is being used this usually results in these spaces being underlined as well. To underline just the text, place a 'white' code followed by a space and then a 'green' code between the centre command and the underline command. For example:

```
<f1>CE<f2> <f1>US<f2>texttexttext
```

TEXT WINDOW DEFINING

The following function key definition will help you to define text windows. Press f0, move the cursor, using the cursor keys, to the bottom left hand corner of the window required, press Return, move to the top right corner, press Return again, and the correct VDU28 statement is printed out.

```
*KEY0X=POS:Y=VP.:G=GET:P=POS:V=VP.:V.31,P,V,46:G=GET:W=POS:Z=VP.:V.31,W,Z,46,31,X,Y
:P."VDU28,";P;",";V;",";W;",";Z|L|M
```

Title : Battlefields
Supplier : BBC Soft
Price : £9.95
Reviewer: Mike Williams
Rating : **

I have just met my Waterloo, or to be more precise, I have been re-playing the battle of Waterloo, one of two campaigns simulated in the new Battlefields package from BBC Soft.

The other simulation included is based on the American Civil War. Both are games for two players, who control the opposing forces, and each game is likely to last as much as two hours. I was unable to load the Civil War game so concentrated on Waterloo, though in general terms both games are similar.

Each player in turn is (secretly) presented with the dispositions of his various commanders and their latest intelligence reports. A map of the battle area is then displayed showing the

Title : Repton
Supplier : Superior Software
Price : £9.95 cass.
£11.95 disc
Reviewer: Alan Webster
Rating : ****

Repton is a multi-level arcade adventure set in an underground cave. The object is to complete all 12 levels without losing any lives.

If you manage to do so, then you are elligible to win the £1ØØ prize for the first to achieve this goal.

The game involves solving a set of problems so that you can collect all of the diamonds within a set time. To hamper you, there are one or more reptiles which, when hatched, chase you to an almost certain death.

The screen displays 1/16th of the playing area of any level. You can move in any one of four directions, and the game

Title : Crypt Capers
Supplier : Software Projects
Price : £7.95
Reviewer: Geoff Bains
Rating : *

Software Projects is a company comparatively new on the Beeb scene making a name for itself by converting its old Spectrum classics to work on the BBC micro. If the company has many more tricks like Crypt Capers up its sleeve, then the name it makes will likely be a less than complimentary one.

This game smacks of conversion. Even the loading instructions are wrong (you try and find the 'earphone socket' on your Beeb!). Games of this standard, though applauded in their time, have long since failed to compete with the best of today's current offerings. It is certainly no way to establish a good reputation.

The action takes place in the Pharo's

Title : Drain Mania
Supplier : Icon Software
Price : £7.95
Reviewer: Alan Webster
Rating : **

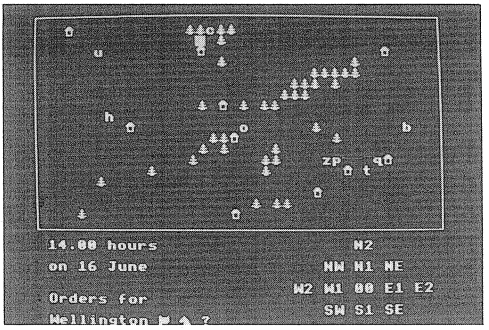
Drain Mania is all about eliminating the horrible bugs, viruses, and other nasties that roam the drains! You control a little man whose task is to rid the underworld of these pests.

Moving the man is a little difficult to get used to as he has a momentum of his own, and does not always respond immediately to changes of direction.

Drain Mania is a multi-level game with a nice interrupt-driven tune which plays while the game is being loaded. There are instructions within the game, and you are allowed to define your own keys. Apart from these touches, the rest of the game is fairly standard.

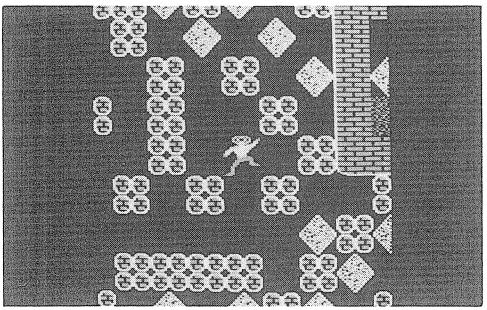
The scenario is a set of platforms

locations of his own forces, and those of any enemy units sighted, and the player then gives his orders to his commanders. Fighting takes place when opposing forces meet, with results based on the relative strengths of the units involved.



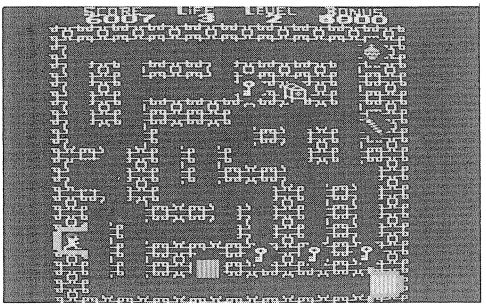
If you're into history and prepared to spend the two hours or so to play one of these games, this package may be of some interest. I felt that a good idea had been lost in a mediocre implementation poorly presented. Oh, by the way, history got it all wrong. In my version Napoleon won!

features an excellent 4-way scroll. The problems set by this game are quite challenging and can be very tricky to puzzle out. A good feature of this game allows you to start on any level, once you have the appropriate password, obtained by completing the previous level.



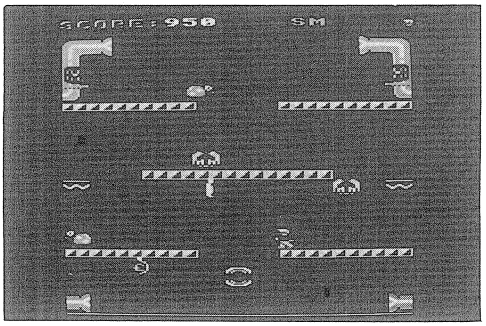
To enter the competition, you must get from level 1 to level 12 in one sitting, and with only four lives, much harder than completing each level separately. Overall, Repton is well worth buying, with the added chance of winning £100.

(their spelling) pyramid. You have to chase around the various maze-like rooms, picking up objects and avoiding or shooting the fireballs, ghosts, snakes, and so on, striving to reach the next chamber - a kind of ancient Egyptian Pacman, but without the cute and cuddly characters.



To say that Crypt Capers is boring is opinionated, but to say that the controls are slow reacting, the backgrounds unimaginative, and sound effects poor is all fact.

with drain pipes at the top and bottom of the screen. You have to 'skate' about the screen, collecting money dropped down the drain while hitting the bugs as they lethargically make their way from top to bottom. After the second screen there is a bonus sheet with lots of money to be collected in a set time.



That description really says it all. This is not one of Icon's better games, it lacks excitement and challenge, and after a few goes has little more to offer even the most avid games player.

BULLETIN BOARDS
The Latest

Peter Rochford brings you up to date with the latest happenings on the communications scene, in particular the growth of bulletin boards using viewdata standards.

Since we last took a look at bulletin boards (BBs) in BEEBUG (Vol.3 No.5), the scene has changed quite markedly. The number of dial-up boards now operating in the UK has risen to well over 100 and the regular updates in the BEEBUG supplement illustrate that this growth is continuing at a steady rate.

What is of great interest is that many of the new boards are now opting for a faster baud rate of 1200/75 instead of 300/300, and use viewdata standards the same as Prestel. This has come about due to the availability of low-cost packages such as the Communitel system, allowing a local viewdata service to be set up on a BBC micro.

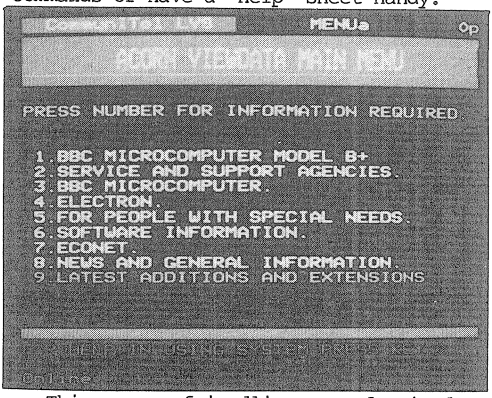
Communitel is available for around £325 and includes the necessary hardware and software for use with a BBC computer and twin 80 track drives. The resultant system can provide 200 frames for retrieval, and has frame editing facilities and the ability to convert files to CET format for telesoftware downloading, the same as used on Prestel and Micronet. The system can be further expanded to give a larger number of frames by using a hard disc.

Pace, well known for the Nightingale modem and Commstar ROM, is, at the time of writing, planning to release bulletin board software, again for the BBC micro. This will be cheaper than the Communitel system but I understand will not be viewdata standard, although it will provide colour.

So it now looks like the Beeb is set to become an important part of the bulletin board scene at the transmitting end of things. Up until now the favourite machine has always been the Tandy TRS 80 model 4, due of course to the fact that this is a popular machine in the USA and most of the software available for running BBs came from there.

Another package that offers viewdata standards is the Metrotel system. This is rather more expensive than Communitel at £1700 and runs on a Torch computer, a machine that has close family ties with the BBC micro.

There is no doubt about it that viewdata, with its colour and graphics, is far more attractive to use than boring old black-and-white. Also, the ease of use of viewdata with its single-key selection of pages from menus makes it very appealing. This is certainly not true of many of the 300 baud systems which are far from user-friendly and require you to remember many commands or have a 'help' sheet handy.



This user-friendliness of viewdata systems coupled with the attractiveness of colour and graphics has made it ideal as a medium for selling services and products. Within the Prestel database we now have homebanking and tele-shopping, plus a host of on-line information services.

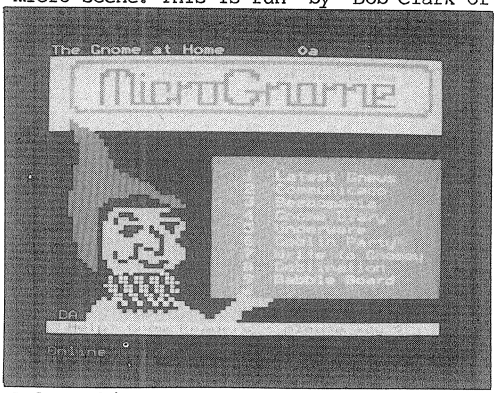
Outside of Prestel, several district councils such as Hackney and Northampton, have allotted areas on their mainframe computers for free access by local residents with viewdata terminals. These show details of council services and amenities, plus information on local events etc.

Beeb owners will be pleased to know that Acorn have at long last set up their own free dial-up information service for

customers. This is a multi-user system running on several BBC micros linked by an Econet file server and using a hard disc for frame storage. The system uses the Communitel software described earlier and shows details of all the products marketed by Acorn Computers. Well worth a look I might add.

Retailers have also seized upon the possibilities of a dial-up information and ordering service. Technomatic, who specialise in BBC micros and peripherals, have just installed their Communitel system and can be accessed 24 hours a day to check the availability of stock and even place orders using credit cards. This can be a great boon to the consumer who can check stock and prices out of shop hours and, of course, releases the staff of a shop from answering price queries on the telephone. I look forward to Watford Electronics with their permanently engaged phone lines, installing theirs (multi-user of course)!

Current users of Prestel and Micronet will probably know already of the Micro-gnome section on the Viewfax 258 pages which features news and gossip about the micro scene. This is run by Bob Clark of



Soft Machinery, a company closely involved with the writing of communications software. Bob Clark has now set up his own database from home called 'The Gnome at Home'. This again uses the Communitel system and Econet file server on several Beebs plus hard disc for storage. This system is shortly going multi-user with 10 lines, and with a further 20 at a later date. Most welcome this, as one of the big problems with private boards is that only one user may access it at one time. There are plans afoot to implement multi-user games on the Gnome board and to include telesoftware too. However, I have a feeling that all these goodies will not be provide free of charge forever, and that a subscription, and advertising, may be used to make the service pay.

There is one section on Prestel that is now becoming extremely popular, and if you haven't looked at it yet then please do. I refer to the Timefame section starting on page 8181. This provides several bulletin boards that deal with specific interests and subjects. You send in your comments and ideas on a response frame and it appears for everyone's viewing several minutes later. This continues all the time the board is open and frames are over-written as new ones are sent. There are many sections for such things as political comment, TV and video, 'Fizz' for graffiti and most important to me, Microboard.

Microboard is dominated I might add by owners of BBC computers and a recent survey quoted 70% of users as Beeb owners. You can put up your questions on any aspect of computing for others to see and you will find that you will always get answers either on Microboard or direct to your Prestel mailbox from other users. Don't ignore this section if you are on Prestel. In fact I would go so far as to say that it is worth the £5 a quarter Prestel charge just to use Timefame.

So there we are. Things are getting very exciting on the comms scene with the growth of viewdata services provided by Prestel and other organisations, and indeed, private individuals. You as a BBC owner are well equipped to take advantage of all this with the Beeb already supporting viewdata standards via its mode 7. All you need is a 1200/75 baud modem, which is becoming increasingly cheap, and some terminal software. Don't wait - communicate!

Phone numbers of UK bulletin boards are featured regularly in the supplement with frequent updates and additions. Viewdata services referred to in the text:

Hackney B.C.	01-985-3322
Northampton B.C.	0604-20441
Acorn	0223-243642
The Gnome at Home	01-348-3274
Technomatic	01-450-9764
Prestel/Micronet	01-618-1111

(Other numbers outside London)

Surac pursues his quest for efficient programming by showing how so-called look-up tables can be used to speed up further a variety of mathematical functions, including those so often used in graphics.

Last month we looked at one way of speeding-up BBC Basic's calculations of sines and cosines. It was much faster than the computer's built-in routines, but only worked when the angles increased at steady rates.

We'll now go on to a way of using tables, or arrays, to speed things up. The approach means that a function's value can be looked up directly, rather than calculating it whenever it is needed. We can use tables to replace either specially written Basic functions or the computer's "intrinsic" functions (e.g. SIN, EXP and LOG).

Let's look at an easy example. Statistical work often needs to calculate factorials (for example 4!=4*3*2*1). It's easy enough to write a function to do the job, but then every one must be calculated whenever it's needed, which could slow things down quite a lot.

A different approach is to fill an array with all possible factorials. The procedure fills the array "Fact(33)" with the factorials of all integers from 0 to 33. By definition, 0! has the value "1", while 33! (value 8.6833E36) is the largest factorial the Beeb can handle. Run the procedure once at the start of your program and then, whenever you need a factorial, use the value from the array. For instance, the number of permutations of "P%" items from a total of "N%" is:

```
Fact(N%)/Fact(N%-P%)
```

As an example of the speed improvement, it takes about 25.5 secs to calculate 500 random factorials directly, while using the table method takes only 1.0 secs, including setting up the table in the first place. The more factorials you need, the greater the advantage, but tables are quicker with as few as 4 calculations.

You are not just limited to your own routines, however - tables can replace, for example, trig functions. It's a little more complex than factorials though, and

FACTORIALS

```
10000 DEF PROCfact
10010 DIM Fact(33)
10020 LOCAL i%
10030 Fact(0)=1
10040 FOR i%=1 TO 33
10050   Fact(i%)=Fact(i%-1)*i%
10060   NEXT
10070 ENDPROC
```

TRIG FUNCTIONS

```
10000 DEF PROCanginit
10010 DIM Sintab(90),Tantab(90)
10020 FOR I%=0 TO 90
10030 Sintab(I%)=SIN(RAD(I%))
10040 Tantab(I%)=TAN(RAD(I%))
10050 NEXT 10060 ENDPROC

11000 DEF FNsin(ang)
11010 LOCAL sign
11020 sign=SGN(ang)
11030 ang=ABS(ang) MOD 360
11040 IF ang>180 THEN sign=-sign:
      ang=ang MOD 180
11050 IF ang>90 THEN ang=180-ang
11060 =Sintab(ang+0.5)*sign

12000 DEF FNcos(ang)
12010 =FNsin(ang+90)

13000 DEF FNtan(ang)
13010 LOCAL sign
13020 sign=SGN(ang)
13030 ang=ABS(ang) MOD 180
13040 IF ang>90 THEN sign=-sign:
      ang=180-ang
13050 =Tantab(ang+0.5)*sign
```

needs a function to extract the data (if all angles are to be catered for) as well as a procedure to set it up.

PROCanginit sets up tables of sines and tangents for single degree steps from 0 to 90 degrees. Once that's done, you can get the trig values for any angle whatsoever by using the fact that trig functions repeat themselves in a totally predictable way. For instance:

```
sin(-A)=-sin(A)
sin(A)=sin(A-360)
sin(A)=-sin(A-180)
```

FNsin uses relationships like these to get the sign and equivalent value, in the range 0 to 90 degrees, of any angle. It then extracts the sine of the equivalent angle from the array Sintab(), negates it if it has to, and returns the value. Calculating cosines is much easier, because cos(A)=sin(A+90).

FNTan works in much the same way, reducing the angle to its equivalent in a fixed range, plus its sign.

These functions do not have the tremendous speed advantage of the factorials, but are still worth using. For instance, 500 random SINs take 12.5 secs, and TANs 22.2 secs, while the equivalent FNsin and FNtan take 7.5 secs and 6.45 secs respectively. Try them in the Polar Curves program from May's BEEBUG (Vol.4 No.1). In addition, by performing most of the calculations first, any subsequent graphics appear much smoother and faster.

If you are doing a lot of trig calculations, savings like this are well worth while but, inevitably, there is a price to pay. The procedures and functions take up memory, while Sintab() and Tantab() between them need over 900 bytes. You must allow for the time to run PROCanginit, so you have to be calculating at least 200 values before there is any overall speed advantage. Finally, the functions only give values for integer angles. Although you can put in fractions of a degree, they are rounded to integers first. For example, FNsin(1.5) actually returns Sin(2.0). This gives a maximum error of around 0.0087, which is normally insignificant.

Why didn't I calculate tangents by using FNsin(ang)/FNcos(ang)? Try it, time it, and see!

When using tables for trig functions, the arrays only need hold a limited range of values, because the functions are cyclic. We can then use this range to get the value for any possible angle. Many other functions are not symmetrical, and we need a different approach.

In such cases, set up the table for the most likely range of values, and use the normal Basic function for values outside that range. For example, suppose we normally need to use EXP on values from -5 to +5, but may have to handle any value.

EXPONENTIAL

```
10000 DEF PROCexpinit
10010 DIM Exp(100)
10020 LOCAL val%
10030 FOR val%=-50 TO 50
10040   Exp(val%+50)=EXP(val%/10)
10050   NEXT
10060 ENDPROC

11000 DEF FNexp(val)
11010 LOCAL ex
11020 IF val<-5 OR val>+5 THEN
        ex=EXP(val) ELSE
        ex=Exp(val*10+50.5)
11030 =ex
```

PROCexpinit sets up a 100-element array for the values of EXP(n) for n between -5 and +5, in 0.1 steps. FNexp returns EXP(val); if "val" is outside the -5 to +5 range, it uses the usual EXP function. Usually, however, it scales "val" to the range 0-100, extracts the value from the table, and returns that. The "50.5" term rounds the index to the nearest 0.1 step to reduce the errors.

You are not, of course, limited to using tables for just these functions, but can use them in many other cases. I've shown you different ways in which you might have to extract the data from the tables and you can pick'n'mix the methods for any particular case. The trick is to identify the range and step-size of the function, fill a suitable array, and write a function to pull out the right data.

Demonstrations of all routines listed are included on the magazine cassette/disc including the modified 'Polar Curves'.

Looking at Data Structures (Part 1)

Sooner or later every programmer encounters problems of how best to store data. Paul Ganney introduces the subject of data structures and describes how to implement some of the more useful techniques in this the first of two articles.

At some point in the construction and development of your programs, you are going to run into a familiar problem: what to do with the data. You may decide to use a FOR-NEXT loop, reading data items one at a time, and use them there and then. But what if you want to use items more than once? What if you don't know the order in which the items will be required? What if you wish to insert new items and delete old ones as the program proceeds? Data structures provide the answers to these and other questions. So let's start by finding out what is meant by the term 'data structure'.

A data structure defines how data is to be stored and how the individual items of data are to be related or located, if at all, in a way that it is convenient for the problem in hand and easy to use. For example, you could hold all your data on index cards and search them yourself for any information you require, but it would be better to hold it all on tape and let a computer do the searching for you. It would be better still to hold it on disc, where an item of data can be accessed ('got at') without having to read all the other items first.

ARRAYS

An array is the simplest form of data structure, and exists as a part of most high level programming languages, BBC Basic included. An array is simply a collection of similar data items, one after another, referred to by the same identifier (variable name). Different items within the array can be referenced by means of a number, called the index, which is an item's position in the array. There is no direct connection between the "value" of a data item and its position.

USING ARRAYS

By using arrays in loops we can often save on programming time and produce shorter programs. This is shown in the following (rather trivial) example which inputs a series of letters and then prints them out in reverse order:

```
 10 REM Array example - P.S. Ganney
 20 CLS:INPUT"How many letters " N%
 30 DIM Letter$(N%)
 40 PRINT'"Input the letters:"'
 50 FOR I%=1 TO N%
 60 Letter$(I%)=GET$
 70 PRINT Letter$(I%)
 80 NEXT I%
 90 PRINT'"In reverse:"'
100 FOR I%=N% TO 1 STEP -1
110 PRINT Letter$(I%)
120 NEXT I%:PRINT
130 END
```

On the BBC micro, the array automatically starts at index 0 (although not used in this example), so that Letter$ actually has N%+1 elements in it.

HIGHER DIMENSIONS

The type of array we've considered so far is known as a one-dimensional array, or vector, but two, three and more dimensions are quite feasible. A two-dimensional array, or matrix, can be thought of as a table with rows and columns. A three-dimensional array can be thought of as data items in a cubic structure. Fourth and higher dimensions are best just used, not worrying about a geometric interpretation.

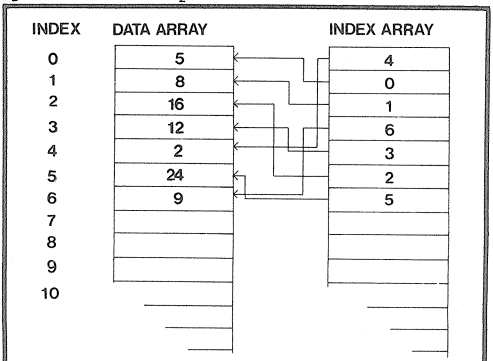


Fig 1 Example of ordered data using an index array

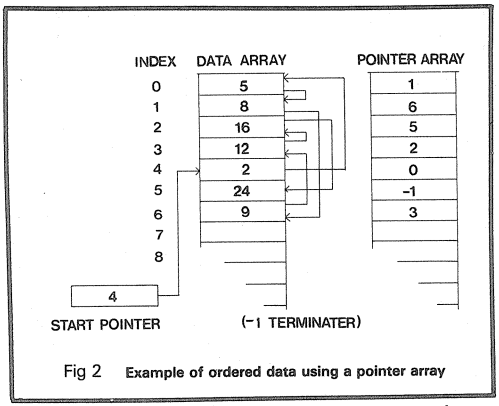


Fig 2 **Example of ordered data using a pointer array**

Higher dimensions allow you to keep related items of data together. For example, a list of prices and a list of corresponding discounts could be held in two arrays, Price(N%) and Discount(N%), or held in a single two-dimensional array, Item(N%,1). Then Item(i%,Ø) would give the price, and Item(i%,1) would give the discount, for the item in position i% in the list. The one point to remember in combining such lists into two (or more) dimensional arrays, is that the data stored must be all of the same type (string, real or integer).

Consider again the three questions posed at the start of this article. Arrays provide answers to the first two, and a partial answer to the third. The problems arise when the items within the array are to be kept in an order. Inserting an item at a specified point in an array necessitates moving all the other items 'below' this point 'down' one location, and deletion involves moving 'up' one location. Consequently, they are often used to hold information that either changes rarely or never alters in order.

INDEXED ARRAYS

Two possible solutions present themselves. Firstly, to provide an "index" array in much the same way as Indexed Sequential Files (see BEEBUG Vol.2 No.4), i.e. each element in the main data array has a corresponding number in the index array, which gives the data item's position. Ordering the data items now involves just ordering the index (as described in more detail in the BEEBUG Workshop for Vol.4 No.1). The idea is shown in principle in Fig. 1. Insertion and deletion here also necessitates a great deal of movement of items, albeit those of the index array only (a great saving if several arrays are being used, one per field (part) of each data item).

For example, if the names, reference numbers and prices of a series of stock items (maximum N%) were to be held in three arrays Stock$(N%), Ref(N%) and Price(N%), then considerable shuffling of data would be involved every time a stock item was added or deleted from the list, assuming that this is kept in order (say, alphabetically by name). This could be reduced by having an index array, arranged in alphabetical order by name, pointing to the data items. Additions and deletions would now only involve moving the index pointers.

We could also have more than one index array, one which keeps the stock items in alphabetical order by name, and one which keeps the stock items in reference number order. This arrangement means that a set of data items can effectively be maintained in two or more orders concurrently with no need for frequent re-ordering. Individual items can be located using the 'Binary Search' technique (see Workshop Vol.4 No.2), in the example searching either by name, or by reference number, as required.

With such a method, insertions are merely added to the ends of the data arrays (or inserted in any free space) and entries inserted into the index arrays at the appropriate points. Similarly, deletions merely require that the index entry (or entries) be deleted (and other pointers moved up). It is not necessary to do anything to the data arrays unless we want to mark the deleted data positions as being re-usable.

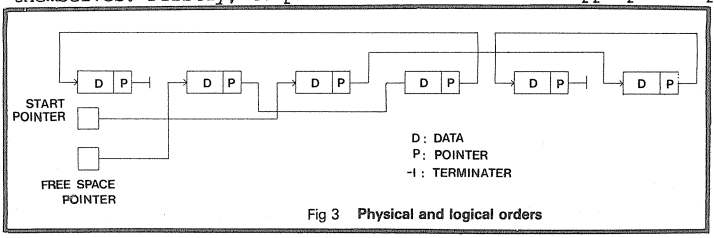


Fig 3 **Physical and logical orders**

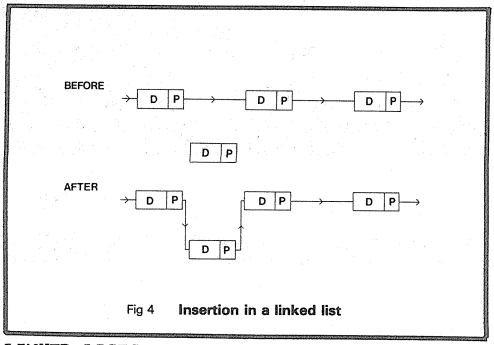


Fig 4 **Insertion in a linked list**

LINKED LISTS

The second method also involves a second array, this time one of 'pointers'. For each element in the data array there is an element in the pointer array which gives the index of the next item in order, thus pointing to it (see Fig. 2). Obviously, this can only work for 1-dimensional arrays (unless the pointer is only needed to point to a row or column). Insertion and deletion is greatly simplified as we shall see. The principles of linking data items by pointers gives a data structure known as a 'linked list'.

With an array structure in its simplest form, the logical structure (which item follows which) is exactly the same as its physical structure (which item is stored after which). This need not always be the case, as Fig. 3 demonstrates. A linked list is an abstract data structure in which it is extremely unlikely that the physical and logical structures will ever be the same (exception: when the list is empty).

Each item in a linked list consists of two parts: the data itself, and a pointer to the next item in the list. In addition, two further pointers must be set up, and a third defined: one to the first item in the list (the start pointer), and one to the next free storage space (the free space pointer). A null pointer (or terminator, i.e. a pointer to nowhere) must be defined, to indicate the end of the list. An empty list is indicated by a terminator in the start pointer, and a full list by a terminator in the free space pointer.

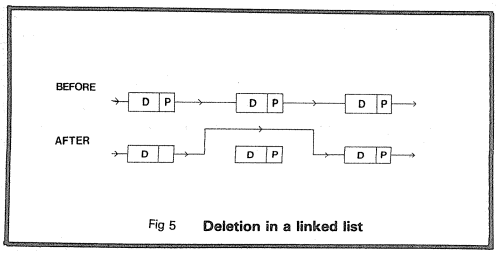


Fig 5 **Deletion in a linked list**

The greatest strength of a linked list is in its flexibility. Inserting and deleting items is incredibly quick and easy to perform. Consequently, linked lists are generally used for applications where data changes often.

USING A LINKED LIST

Insertion or deletion is simply a matter of adjusting pointers (see Fig. 4 and Fig. 5 respectively, but note that these diagrams represent the logical rather than physical structure of the list). After a few insertions and deletions, the list will spread through store, with deleted items lying physically between items still in the list. These are collected together to form another linked list, of free data items. Storage space for new items is taken from this free space list, and that gained from deletions is given to it.

Returning to our previous example of stock items, we have a further complication if we want to use linked lists, as we want to link the data items in two different sequences, one alphabetically by name and one in reference number order. We can do this by associating two pointers, called P1 and P2, with each set of data, as in Fig. 6, where P1 pointers link the items in alphabetical order and P2 pointers link the items in reference number order. Two start pointers are now needed, and the free space list can use either pointer (in this case P1 is chosen).

[To be continued next issue.]

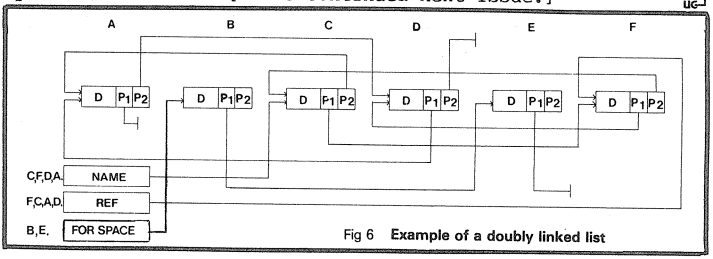
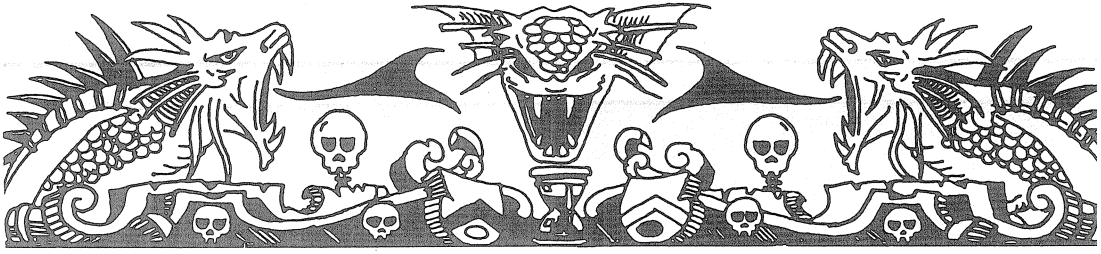


Fig 6 **Example of a doubly linked list**



by Mitch

ADVENTURE GAMES ADVENTURE GAMES

Swimming around in my haunted fish-tank this month are two text adventures from Robico Software.

ISLAND OF XAAN from Robico Software, 3 Fairland Close, Llantrisant, Mid Glamorgan. Cassette £7.95 Disc £9.95.

As the mist cleared I found myself in a windowless dungeon without so much as a rat for company. With more than a little difficulty, I finally emerged to find a slumbering guard who could teach my dragon a thing or to about light sleeping. Any movement on my part resulted in him launching himself from his cot and separating my head from my shoulders.

```
>LOOK

You are in a small, dimly lit prison
cell, filled with the stench of death
and decay. The smooth, granite walls
seem to close in on you, windowless
and menacing! To the north looms a
huge, iron door.
The door is closed. The only visible
exit is north.
You find an alcove.

>EXAMINE CHAIN

The chain is fashioned from iron, its
heavy links encrusted with rust from
ages past. You find nothing.

>EXAMINE ALCOVE

The alcove is nothing special. You
find nothing.

>_
```

When, and if, you manage to pass this kindly soul you will find the many paths which criss-cross the island. Xaan is littered with jewels, shovels, beggars and the Hog in the Dark - which bites! The game accepts two word commands which will lead you through 180 locations before you find enough treasure to buy yourself a passage to freedom.

This is a traditional adventure which is very similar in style and format to the early EPIC range (e.g. Castle Frankenstein). Acres of text have been compressed into this game and used to good effect. A teleprinter, which is to be found in the middle of a deserted plain, curtly informed me to go away as it is sitting there feeling surrealistic. Hands up all those who have never heard of Dali!

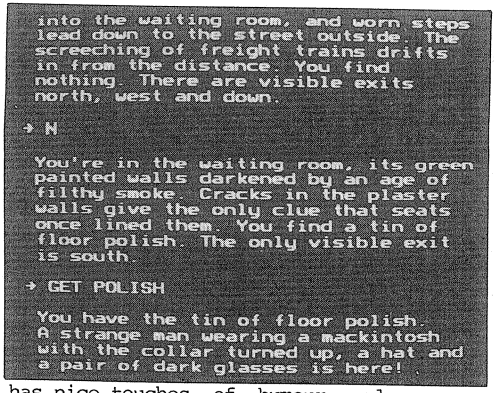
I have returned to this game every night for a week, convinced that tonight I will find the limits of the island, and so far I am still mapping! What began as an irritating game is becoming an obsession as I am beginning to realise how much has been shoe-horned in. I have found so many objects to store and manipulate I'm becoming bemused. This game certainly has hidden depths in which you will wander for many an hour. Very professional...

ASSASSIN also from Robico Software. Cassette £9.95 Disc £11.95.

This adventure takes place in a world of intrigue, double-cross and assassination - not unlike the BEEBUG office!

Quickly slipping into my gumshoes and dark glasses, I entered the game by the side entrance and merged with the shadows. It appears that my mission is to blow away some wise guy who is causing trouble for the front office. As my previous contract was as a hit-man for the Brownies, this job should be a sneeze!

The trail leads through 200 locations of railway stations, mountain villages, snowy peaks and army camps, all filled with the bloody remains of previous assassins. The concept of time passing is handled nicely, with some events being in a different state should you return after an overnight sleep. To find and follow the clues left for you by your contact, involves some code-breaking and neat timing, but it's all great fun. The game



has nice touches of humour and a very professional feel in all of its routines. Full sentence commands are accepted, which is a mixed blessing, as there is one occasion when I knew what the answer was, but I could not phrase it to the game's satisfaction. What I should stress is that I found the game one of the best I've played for many a month. The problems are of average complexity, which made it fun rather than a task, and kept me playing into the early hours. Highly recommended.

This month's spell comes from Mark Smith of Rochester and it shows how to disguise the true end of your program.

The LIST command will halt naturally whenever it encounters the hex characters &ØD and &FF. If we insert these at a point which looks like a normal end, the casual user will look no further.

```
10 PRINT "MAIN PROG"
20 PROCFRED
30 END<space><space>
40 DEFPROCFRED
50 PRINT "MAGIC"
60 ENDPROC
```

Type in the above, adding two space characters (represented as <space><space>) behind the END command at line 30. Now find these two spaces (&20, &20) in memory and change them to &ØD and &FF (e.g. ?&E1D=&ØD :?&E1E=&FF). The program will continue to work as normal but a LIST command will halt innocently on line 30.

POINTS ARISING

TRACKMAN (BEEBUG Vol.4 No.2)

A mistake during the production of this program resulted in three lines of this program appearing incorrectly. Fortunately, these lines only affected the initial display of instructions and the score, and did not upset the game itself. We are sorry for any confusion that may have arisen over this. The lines should read as follows:

```
2700 PRINT"£";TAB(12);LEVEL;TAB(22);L;TAB(26);:IF BL=1 PRINT TIME;SPC(2) ELSE PRINT TTG-TIME;CHR$32
2770 PRINTTAB(8,7)"By Tom and Silas Standage":COLOUR3
2780 PRINT''" Guide the train(s) safely around the"'" track, until the timer reaches zero."'" Then shunt each train back to the"'" station as quickly as possible."
```

BEGINNERS START HERE - LOGIC (BEEBUG Vol.4 No.2)

In one of the small examples on page 22, the line C=C+(A=3) should read C=C-(A=3).

SPREADSHEET PROGRAM (BEEBUG Vol.3 Nos.9 & 10)

Two small problems have emerged concerning this program. Basic I users should replace 'OPENUP' in line 4230 by 'OPENIN'. These both have the same token in Basic, so that a program entered with 'OPENUP' using Basic II will then work correctly if used with Basic I. Secondly, the test for division by zero does not work correctly if a row or column is being divided by a constant. Replace lines 5470 and 5510 as follows:

```
5470 IF EVAL("mat(D1%"+Z1$+",D2%)") AND Z$="/" THEN mat(A1%+I%,A2%)=0:GOTO5490
5510 IF EVAL("mat(D1%,D2%"+Z1$+")") AND Z$="/" THEN mat(A1%,A2%+I%)=0:GOTO5530
```

CROSS REFERENCER (BEEBUG Vol.3 No.6 and Vol.4 No.1)

The author of this program has confirmed an error (despite testing) in line 1860 published in his update to this useful utility. The line should commence:

```
1860 UNTIL i%>eline% . . . . . .
```

and not as printed in the update. Thanks to Mr.Coleman for spotting this.

The BEEB'S Disc Catalogue Revealed

James Fletcher encourages you to dig into your disc catalogue and find a whole mine of information, of interest to both new and more experienced disc users alike.

One of the beauties of the BBC micro's disc filing system, as far as beginners are concerned, is that it is very easy to use even if you don't know how it works. Just by following a few simple commands the user can rapidly load and save programs. How convenient it is, to be able to call up an on-screen catalogue, is one of the first things noticed by someone who has only previously used cassette. This gives details of all the programs on a disc merely by typing "*CAT" or even "*.".

People who use public libraries are usually aware that a catalogue of all the available books exists, but only if they take the trouble to find out how the cataloguing system works will they be able to use the system to its maximum advantage - being able to sort out all the books by a particular author or on a particular subject, for example. In the same way, the beginner can straightaway use the Beeb's disc catalogue for basic information. However, finding out just a little more about how the system works will pay big dividends by providing a great deal of extra information. This can be invaluable to the user who is keen to make the most of his DFS. This article will help you to delve into the 'innards' of the DFS catalogue system, and show you how to make use of all the information that is contained in the first few sectors of the floppy disc.

When you take a new disc out of its wrapping, the surface coating is just an unstructured jumble of microscopically fine iron-oxide particles, and if you try putting it into your disc drive and typing "*CAT", your computer will try in vain to make sense of the noise signals picked up by the drive's read/write head. Since there are no signals present recognisable as computer data, it will eventually give up and provide a suitable error message. Before you can use the disc for data storage you have to 'format' it, a process that records 40 or 80 magnetic concentric circles on the disc and provides magnetic marker pulses which divide each of these circles into 10 sectors.

Tracks are numbered from 0, on the outside of the disc, to 39 or 79 on the inside, depending on whether your drive can use 40 or 80 track discs. Sectors are numbered from 0 to 9, sometimes shown as 00 to 09, sector 0 occurring shortly after the tiny 'index-hole' that you can see punched near the centre of the disc. Every time the disc rotates the index hole will allow light from a light-emitting diode to pass through to a photo-detector, and when this pulse is detected the disc controlling system knows that the head is positioned over sector 0, the start of a track. Extra magnetic markers are laid down during the formatting process to identify the first two sectors of the disc as the catalogue, and various other synchronising, addressing and error-checking pulses are also added at this time.

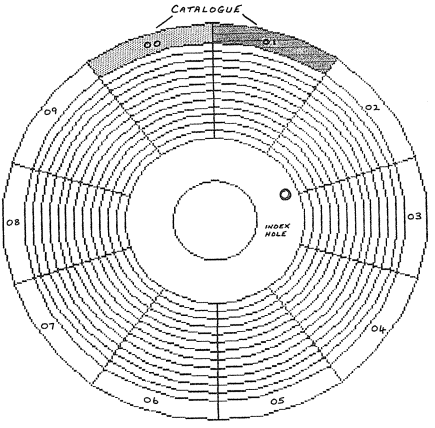


Fig.1. Disc layout showing catalogue on sectors 00 and 01 of track 0

Since it is the first two sectors, 0 and 1, on track 0 that form the disc catalogue, we will take a look at exactly how these are made up, and then show how to use the information that they contain. Like all the other sectors on the disc the first two each contain about 300 eight-bit bytes (think of a byte as a character), some being used for so-called 'housekeeping' functions, like synchronising the rotation of the disc with the pulses from the control chip, for identifying the

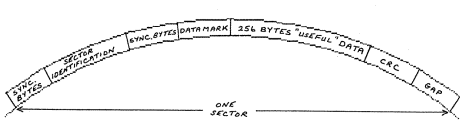


Fig.2. Layout of a typical sector showing 256 bytes of "Useful" data plus 'Housekeeping' information

SYNC – Synchronisation information
CRC – Cyclic Redundancy Check, a checking code
DATA MARK – Says if data exists or has been deleted

number of the sector, and for checking that no read/write errors have occurred. The remaining 256 bytes are available to carry useful information, and it is these data bytes that we shall concentrate on.

If you "*CAT" a disc, the screen shows something like this :

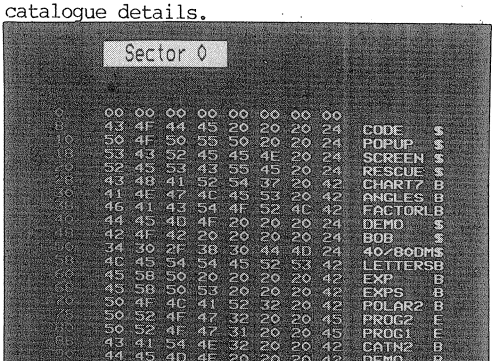
```
TITLEOFDISCS (29)
Drive: 0                Option: 3 (EXEC)
Directory:0.$           Library :0.$

  !BOOT                 BEEBUG1
  COLOURS               DREDFUL
  EGGCUP                FASTER
  FLAG                  GOODBYE
  PIGLET                RAPIDS
  YANKEE                ZEBRAS

A.FISHES L          B.TESTER
```

The top line contains the title of the disc, TITLEOFDISCS in our case (set with the *TITLE command). The number in brackets following the title is the number of times that the disc has been written to, and can be useful for tracing which version of a program you are using. If a disc is re-formatted, this number returns to zero. The second line contains the drive number, effectively telling you which side of the disc you are using, and shows which of the various start-up options the disc is set for (see page 50 of the Acorn Disc System User Guide). Our example shows option 3, (which has been set by typing *OPT4,3) and this means that with the appropriate !BOOT file on the disc, pressing Shift/Break can be made to automatically load and run a program from the disc.

The third line of the header shows the currently selected directory, set, as in our example, to $ (the default directory) unless you choose another character (by typing *DIR followed by another letter). This is followed by the currently selected 'Library', in our case drive 0 and directory $. Machine code programs which are contained in the library can be loaded and run merely by typing *filename. Then follows a complete list of all the files on the disc, in alphabetical order. Files in the currently set directory ($) are listed first, followed by those in other directories, in our case 'FISHES' in directory 'A' and 'TESTER' in directory 'B'. Note that 'FISHES' is followed by a letter L, showing that the file is locked against accidental erasure.

All this catalogue information is fitted into the 256 useful bytes of each of the first two disc sectors as shown in Table 1 overleaf.

Typing in the short program listed at the end will enable you to prove for yourself that the catalogue information given in the table is correct, and will let you see how the first two sectors on any disc are made to contain all the catalogue details.



Line 110 reserves two sections of memory. One is called 'buffer' and is 11 bytes long, the other is called 'sector' and is 256 bytes. Lines 120 to 300 make

```
=============================================================
SECTOR Ø   SYNCHRONISATION,IDENTIFICATION,& CHECKING BYTES
=============================================================
   BYTE &ØØ..... T
   BYTE &Ø1..... I
   BYTE &Ø2..... T          First eight bytes of
   BYTE &Ø3..... L          twelve byte (character)
   BYTE &Ø4..... E          disc title. Note that last
   BYTE &Ø5..... O          four bytes of title are in
   BYTE &Ø6..... F          next sector.
   BYTE &Ø7..... D
   BYTE &Ø8..... Z
   BYTE &Ø9..... E
   BYTE &ØA..... B          Up to 7 bytes (characters) for
   BYTE &ØB..... R          name of first file on disc.
   BYTE &ØC..... A
   BYTE &ØD..... S
   BYTE &ØE.....
   BYTE &ØF..... $          Directory of first file on disc.
   BYTE &1Ø..... B
   BYTE &11..... E
   BYTE &12..... E          Up to 7 bytes for name of
   BYTE &13..... B          second file on disc.
   BYTE &14..... U
   BYTE &15..... G
   BYTE &16..... 1
   BYTE &17..... $          Directory of second file.
   BYTE &18-BYTE &1E        Name of 3rd file (7 characters)
   BYTE &1F..... $          Directory of 3rd file.
   BYTE &20-BYTE &26        Name of 4th file.
   BYTE &27..... $          Directory of 4th file
   ..... to BYTE &FF        and so on for up to 31 files.
=============================================================
SECTOR ØØ END     CHECKING BYTES FOLLOWED BY GAP
=============================================================
=============================================================
SECTOR Ø1          SYNCHRONISATION AND CHECKING BYTES
=============================================================
   BYTE &ØØ..... I
   BYTE &Ø1..... S          Last  four bytes of
   BYTE &Ø2..... C          disc title.
   BYTE &Ø3..... S
   BYTE &Ø4..... 1D         Number of times disc written to.
   BYTE &Ø5..... 7Ø         8 times No. of catalogue entries.
   BYTE &Ø6..... bit Ø - Number of sectors on disc (bit 1)
                 bit 1 -  "     "       "     "  "  (bit 2)
                 bit 4 - Start-up option (bit 1)
                 bit 5 -  "      "       " (bit 2)
   BYTE &Ø7.....          Number of sectors on disc
   BYTE &Ø8.....            (bits 3 to 8).
   ..... to BYTE &ØF      Data about first file. Load,
                            Execution and Start sector
                            addresses plus file length.
   BYTE &1Ø.....            Information about second file,
   ..... to BYTE &17      as above.

Repeated in blocks of eight bytes for up to thirty-one files.
=============================================================
SECTOR Ø1 END     CHECKING BYTES FOLLOWED BY GAP
=============================================================
```

use of one of the Beeb's built in routines for handling disc information. By using this OSWORD command (line 290) with the accumulator set to &7F (line 260) the computer is told to perform a read/write operation in which it reads sector Ø from disc and puts this into the area of memory that was reserved as 'sector'.

In lines 150 to 250 the parameters for the OSWORD call are inserted into 'buffer'. A memory address offset is used here, so that in line 160, for example, we are putting information into 'buffer' one byte from the beginning, or in other words, into byte 2. Note that line 130 contains the drive number, and you can change this from Ø to 1, 2, or 3 if you wish to see what is happening on your other drives.

The contents of 'buffer' are set as shown:

byte Ø Drive number (Ø).
byte 1 Low and high byte
byte 2 of 16 bit 'sector' buffer address.
byte 3 Extended address
byte 4 bytes (both Ø).
byte 5 Number of parameters (3).
byte 6 &53 for READ, or &4B for WRITE.
byte 7 Track number (Ø).
byte 8 Sector number (Ø).
byte 9 &2Ø (sector length 256) + number of sectors (1)

The track and sector numbers above are calculated from the value of S%, initially Ø (see also below).

Once the disc information is stored safely in the computer using the reserved memory area 'sector', and in the format detailed in Table 1, line 330 starts the main program loop and reads bytes out of 'sector' in groups of eight. Lines 360 to 390 form another loop, this time printing out the information a byte at a time (in hexadecimal), and then as a string of characters. After sector 0, the whole process is repeated for sector 1. The one procedure, at the end of the program, provides a header for each sector display, giving track and sector number. Note that you will need to press Shift to scroll each pageful of information up the screen.

After saving the program, running it should give you a complete display of the contents of the disc catalogue, and armed with the information in the first part of this article you will be able to browse at leisure through all the detail that the catalogue contains. You can also alter the limits of the loop controlled by S% (line 140). This will allow you to look at other sectors on any disc. The value of S% can range from 0 for the first sector up to a maximum of 399 (40 track) or 799 (80 track).

```
 10 REM DISC CATALOG PROGRAM
 20 REM Version B1.1
 30 REM Author Neil Slater
 40 REM BEEBUG Aug/Sept 1985
 50 REM Program subject to copyright
 60 :
100 MODE 7:VDU14
110 DIM buffer 11,sector 256
120 osword=&FFF1
130 drive=0
140 FOR S%=0 TO 1
150 ?buffer=drive
160 buffer?1=sector MOD 256
170 buffer?2=sector DIV 256
180 buffer?3=0
190 buffer?4=0
200 buffer?5=3
210 buffer?6=&53
220 buffer?7=S% DIV 10
230 buffer?8=S% MOD 10
240 buffer?9=&21
250 buffer?10=0
260 A%=&7F
270 X%=buffer MOD 256
280 Y%=buffer DIV 256
290 CALL osword
300 IF buffer?10<>0 THEN PRINT"Error."
:END
310 PROCprintsector(S%)
320 PRINT'CHR$129;"Byte"
330 FOR J%=0 TO 255 STEP 8
340 PRINT'CHR$130;~(J% AND 255);
350 PRINTTAB(5);CHR$134;
360 FOR I%=J% TO J%+7
370 BYTE=sector?I%
380 IF BYTE<16 THEN PRINTTAB((I%-J%)*3
+6);"0";~BYTE;" "; ELSE PRINTTAB((I%-J%)
*3+6);~BYTE;" ";
390 NEXT I%
400 VDU131
410 FOR I%=J% TO J%+7
420 BYTE=sector?I%
430 IF BYTE>31 AND BYTE<127 THEN VDU B
YTE ELSE VDU32
440 NEXT I%,J%
450 PRINT''
460 NEXT S%
470 END
480 :
1000 DEF PROCprintsector(S%)
1010 LOCAL sector$,track$
1020 sector$="Sector "+STR$(S%MOD10)
1030 track$="Track "+STR$(S%DIV10)
1040 FOR Y%=0 TO 1
1050 PRINTTAB(5);CHR$141;CHR$157;CHR$13
2;track$;SPC4;sector$;SPC2;CHR$156
1060 NEXT Y%
1070 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS

BASIC LINE ADDRESSES - Dave Brinicombe

This short function key definition prints out the start memory address, line number, and length of each line of a Basic program.

```
*KEY0CLS:@%=6:A=PAGE:P."PAGE ";~A:REPEAT P.~A," line ";256*A?1+A?2,"length ";A?3:
A=A+A?3:UNTIL A?1>&7F:P."TOP  ";~TOP|M
```

OS EXTINCTION - Dave Brinicombe

Not exactly a hint, but...the fate of the famous species of flightless bird of Mauritius is revealed by disassembling the OS ROM from location D0D0 (!):

```
D0D0   20 ED D1   JSR D1ED
```

PLOTMATE

Plotmate is another recent product offering high quality graphics output. Geoff Bains has been putting the plotter through its paces.

When you have completed your latest graphics masterpiece on the screen, there is nothing better than a copy on paper to show your friends. The only ways of achieving hard copy of graphics are either to use a printer dump routine with a normal dot matrix printer (not the best method of displaying a colourful picture) or to use a plotter. The latter produces excellent results but at a cost of £500 or more. Linear Graphics' Plotmate aims to change all this. Capable of high quality output, the Plotmate is a flatbed A4 plotter with driving software especially for the Beeb, all for £344.

Plotmate is a very simple piece of machinery. Housed in a rugged metal case, it comprises a sloping surface to take your paper, an arm that moves across the paper (held in place with magnetic strips), and a pen holder that moves up and down along the arm. The whole thing is controlled by two stepper motors with no 'intelligent' electronics to be found.

The secret of Plotmate's success is in its software. Designed specifically for the Beeb, the driving software intercepts normal graphics commands (such as PLOT and DRAW, and machine code calls to the OS), and translates them into control signals for the plotter's stepper motors. Even commercial packages (such as BEEBUGSOFT's Hershey Characters) produce good results.

This means that using Plotmate is as easy as using your BBC micro's screen. Once you have written the program to produce the picture, graph, or whatever, on the screen, the Plotmate software (on disc) is used to produce the picture on paper. There is no plotter graphics language to learn.

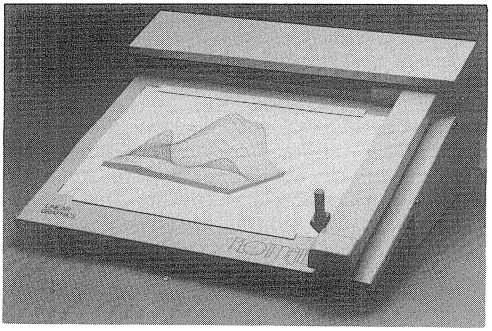
The Plotmate can only handle one colour at a time (it uses standard plotter pens available from many dealers, or direct from Linear Graphics at about £7.50 for six). When your program demands a change of colour (a VDU18 or a GCOL command) the plotting stops to allow you to change the pen - a simple job with the bayonet mounting.

Plotmate can handle characters too. Letters appear on the paper differently to those on the screen because they are built up from lines rather than pixels. Although user defined characters cannot be reproduced, VDU23 is used to alter the orientation of printable characters and their size on paper (anything from a few millimetres to simply enormous).

The entire size of the plot can be altered and the orientation too. This means that although the plotter can only handle an A4 sized section at a time, a plot up to A2 can be produced by drawing a quarter at a time.

Perhaps the most amazing feat that Plotmate can perform is shading. Although large areas are filled by drawing a series of lines on the paper next to one another, the results are very dark and even. This really demonstrates the accuracy to which Plotmate has been made and is capable of being operated. Indeed, its resolution far exceeds that of the Beeb's display in any mode. Another addition is its ability to vary the pattern used to shade areas when a PLOT85 type command is used.

Plotmate is not cheap, but at less than £350 it is cheap for a flatbed plotter. The results are not inferior either. As a plotter for the Beeb, Plotmate is both versatile and hardwearing. If you have the cash it is recommended.



6502 DEVELOPMENT PACK

The long awaited 6502 development pack from Acornsoft has at last appeared. Donald Morgan brushes the cobwebs off his second processor and reports on the package.

6502 Development Pack produced by Acornsoft and priced at £49.85 inc. VAT.

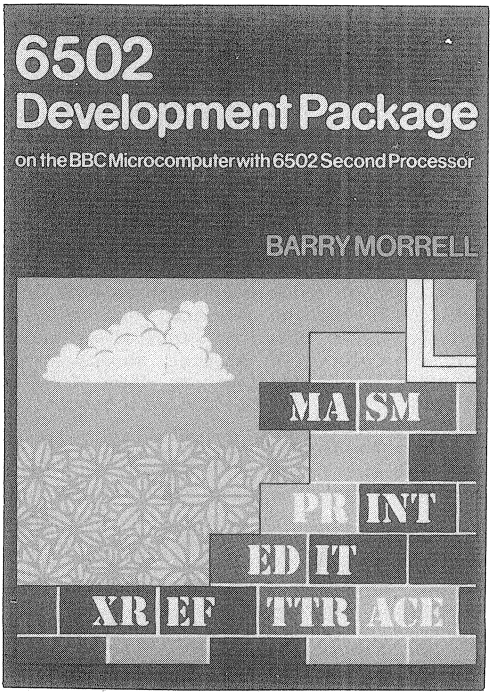
One of the nice features of the BBC micro is its built-in assembler. Although quite adequate for the beginner, is this assembler really suitable for serious program development? A number of manufacturers have already given their answers to this question in the form of stand alone assemblers (for example, see VASM review Vol.3 No.1). Now, a year and a half after the launch of the 6502 Second Processor, Acornsoft have released an assembly package, designed specifically for the 6502 Second Processor, that takes advantage of the extra speed and memory of this device. This is the 6502 Development Pack, or DevPack for short.

Individual programs within this package include a truly superb text editor, a powerful macro assembler, a cryptic debugger (more on that later), a print utility and a cross referencer. To run DevPack you will require both a 6502 Second Processor (at about £200) and a disc drive (dual drives are preferable). The manual assumes that you already have a fair knowledge of 6502 assembler; although clear, it would provide hazardous reading for the absolute beginner.

The text editor supplied with the DevPack is very powerful. The reference card, for example, gives a 7 keystroke sequence to reverse alternate characters within a document. Not many editors offer that degree of flexibility in their search and replace options. Control codes (if desired) are catered for, with the maximum size file being approximately 47k. This is a well thought out editor, extremely pleasant to use and difficult to fault.

MASM (Macro ASseMbler) is the main program in this package and has been written specifically for a 6502 Second Processor. It assembles code very fast thanks to efficient buffering using the extra memory available. The macros that it offers can lead, via macro libraries, to the creation of a pseudo high level language. Local and global labels, conditional assembly, looping and recursion, file linking and full macro parameter substitution all go to make MASM a very powerful assembler. MASM provides some good numeric and string manipulation operations - credit is due here, as they are comprehensive and carefully thought out. Unfortunately, MASM does have some serious flaws.

Instead of using the more normal (and BBC 'standard') MOS Technology assembler mnemonics, MASM uses the obscure Rockwell versions. This leads to LDA #4 being written as LDAIM 4, for example. The two short listings below assemble identical code (the first in BBC Basic format and the second in MASM) and should serve to indicate MASM's approach to key aspects of layout, etc. The mnemonics do have one advantage - the extra speed that results from including addressing information in the instruction code.



When considering whether you should use MASM, one of the main decisions will be whether or not you would use these mnemonics. Personally, I am prepared to change, but I know a couple of major BBC authors who say they would not change over to MASM to assemble their programs.

Another weak point with MASM is the restriction on label length - six characters! If local labels were not available this would have been a pretty damning flaw.

By means of the 'CPU' directive, MASM allows the additional CMOS op-codes to be assembled as well as standard op-codes. Although this is a nice feature, there is very little available in the way of detailed documentation on these new op-codes. The otherwise excellent manual would have benefitted from the inclusion of some extra material to enlighten the newcomer to these extensions to the 6502 instruction set.

To aid the user in the inevitable debugging process, Acorn have included a program called Trace, which, to its credit, offers most of the facilities that one might expect. These include 'simulation' with range selection, breakpoints, memory protection, display, editing, etc. The problem with Trace, though, is the user interface. Commands are all two letter entries with some very cryptic derivations. Memory has been termed store and disassembly is renamed as store interpreting - hence (I think) 'IT' is the command for disassembling memory. Numbers in Trace are entered in hexadecimal (no other base is allowed) but are illegal if not preceded by an ampersand. I couldn't find a logical reason for this and would have thought that the ampersand should not have been necessary.

Apart from the cryptic two letter commands and the insistence upon a preceeding ampersand, my other criticism of Trace is that it cannot disassemble CMOS op-codes (not mentioned in the manual, incidentally). This means that Trace is not capable of coping with everything that one might assemble with MASM. The disassembly process does not correctly adjust branches to their final destination either, but just prints the offset. Neither could I find any way to display memory in ASCII. This program is, frankly, a disappointment.

BBC Basic Assembler Listing

```
 10 REM Example
 20 DIM CODE 40
 30 oswrch=&FFEE
 40 FOR PASS=0 TO 2 STEP 2
 50   P%=CODE
 60   [OPT PASS
 70   .start
 80   LDX #32
 90   .loop
100   TXA
110   JSR oswrch
120   INX
130   CPX #127
140   BNE loop
150   RTS
160   ]
170 NEXT
```

MASM Listing

```
; > Example
oswrch  *       &FFEE
        ORG     &2000
start   LDXIM   32
loop    TXA
        JSR     oswrch
        INX
        CPXIM   127
        BNE     loop
        RTS
        END
```

There are some additional utilities with the package. The first of these is a print program which will generate source listings with more extensive formatting than is available within the assembler. A second version of this program works without a second processor attached. There is also a cross referencer, again in two versions.

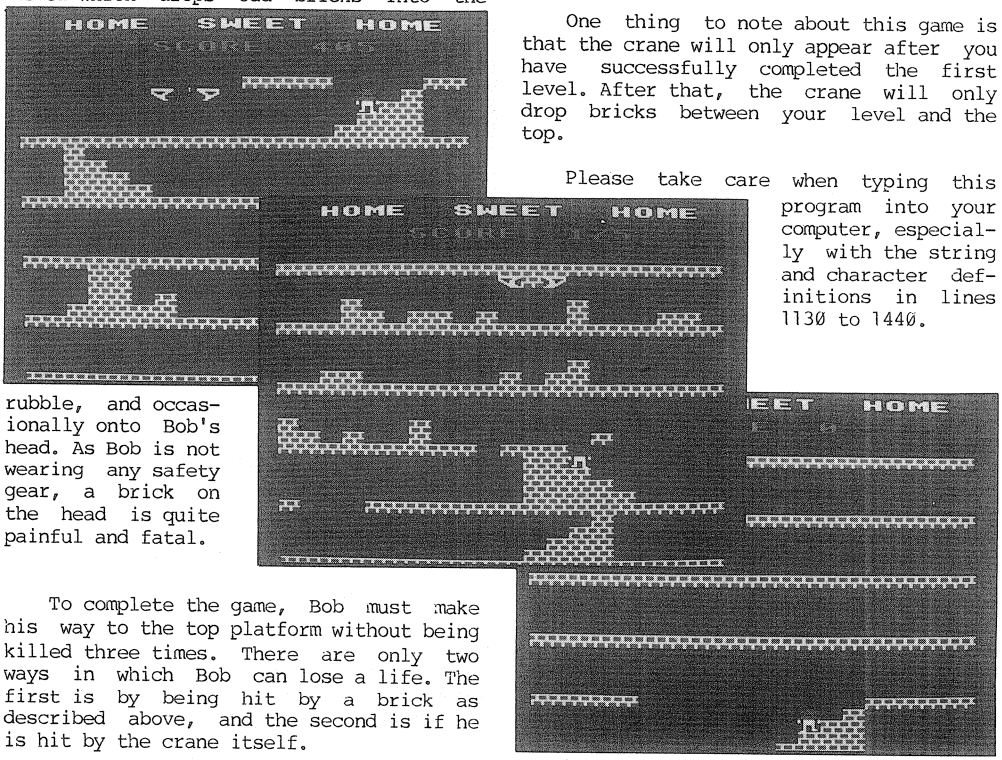
Overall, this is a very powerful package (probably the most advanced of its kind for the BBC micro), and can be recommended if the weak points mentioned above are borne in mind. Personally, I think the good points in this package outweigh the bad ones, and suggest that all serious programmers give every consideration to this package.

BUILDER BOB

Richard Lewis, author of last month's fast moving Hatrace game, has dreamed up another all-action game. Can you help Builder Bob to build his escape in this challenging game of speed and tactics?

Bob is trapped on the bottom floor of a building that is being demolished. Will Bob be able to escape with your guidance or will he become just another brick in the wall?

Builder Bob is a fast action one player game of skill and strategy, where you have to pile up bricks to help Bob build his way out of the condemned building. This all sounds easy, until you meet the workman's crane on the second level which drops odd bricks into the rubble, and occasionally onto Bob's head. As Bob is not wearing any safety gear, a brick on the head is quite painful and fatal.



To complete the game, Bob must make his way to the top platform without being killed three times. There are only two ways in which Bob can lose a life. The first is by being hit by a brick as described above, and the second is if he is hit by the crane itself.

The keys for playing are included in the programs title page. They are as follows:

Z and X move Bob left and right respectively, > and ? pick and drop bricks from Bobs left or right and the space bar makes him jump to take or replace a brick from above.

PROGRAM NOTES

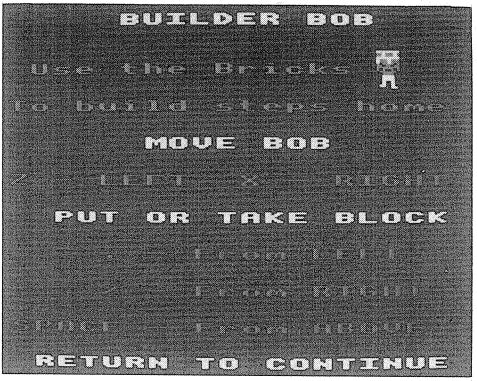
There are not a lot of procedures in this program so we have outlined each one below.

PROCinit	Initialise variables
PROCmanch	Define characters for man
PROCmake	Define strings for characters
PROCman	Move and test man
PROCinst	Instructions
PROCscore	Print score sheet
PROCcrane	Position and move crane
PROCcheck	Test for Bob being hit by a crane
PROCcheck2	Test for Bob being hit by a brick

One thing to note about this game is that the crane will only appear after you have successfully completed the first level. After that, the crane will only drop bricks between your level and the top.

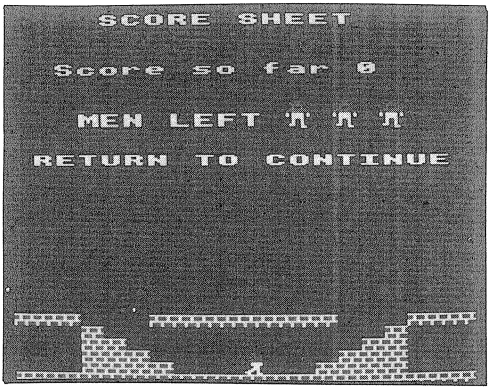
Please take care when typing this program into your computer, especially with the string and character definitions in lines 1130 to 1440.

```
   10 REM PROGRAM BUILDER BOB
   20 REM VERSION B0.3
   30 REM AUTHOR  R. LEWIS
   40 REM BEEBUG  AUG/SEPT 1985
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 ON ERROR GOTO 200
  110 MODE5:VDU23,1,0;0;0;0;:VDU19,3,5,0
,0,0:D%=0
  120 DIM C$(6),A%300
  130 ?&D0=2:?A%=0:A%=A%+1:A%?147=10:HSC
%=0:VDU23,1,0;0;0;0;:VDU23,242,0,221,221
,221,0,119,119,119:F%=0
  140 PROCinst
  150 PROCscore:PROCinit
  160 REPEAT:PROCman:PROCcrane:UNTIL F%>
4 OR ER%
  170 FOR I%=1TO4:PROCcrane:NEXT:IF MEN%
<1 OR F%>4 THEN PROCscore:MEN%=3:SC%=0
  180 GOTO150
  190 :
  200 ON ERROR OFF
  210 MODE 7:?&D0=0:IF ERR=17 END
  220 REPORT:PRINT " at line ";ERL
  230 END
  240 :
 1000 DEF PROCinit
 1010 CLS:COLOUR3:PRINTTAB(2,1)"HOME  SW
EET  HOME";:COLOUR1:PRINTTAB(6,3)"SCORE
 ";SC%;"   ";
 1020 COLOUR130:FORJ%=1TO6:FOR I%=0TO19
 1030 PRINTTAB(I%,J%*5+1);CHR$242;
 1040 K%=I%+1+J%*21+A%:?K%=1:IF J%=1 THE
N K%?-21=0:NEXT ELSE NEXT
 1050 K%=J%*21+A%:?K%=10:NEXT
 1060 COLOUR128:CP%=0:BD%=0:TA%=0:ER%=FA
LSE:BL%=0
 1070 VDU23,243,129,255,129,129,129,255,
129,129
 1080 VDU23,244,255,255,134,204,108,60,3
0,15
 1090 VDU23,245,255,255,97,51,54,60,120,
240
 1100 M%=0:IM%=0:IB%=0:IX%=0:X%=0:IY%=0:
Y%=30-5*F%:V%=0:B%=0:PROCmanch:PROCmake:
E$=E2$:M$=M2$
 1110 ENDPROC
 1120 :
 1130 DEF PROCmanch
 1140 ON 2*M%+IM%+1 GOTO 1150,1160,1170,
1180,1190,1200,1150,1160
 1150 IF B%=0 THEN VDU23,230,189,189,60,
36,36,36,100,6:GOTO 1210 ELSE VDU23,230,
60,60,60,36,36,36,100,6:GOTO 1210
 1160 IF B%=0 THEN VDU23,230,189,189,60,
36,36,36,38,96:GOTO1210 ELSE VDU23,230,6
0,60,60,36,36,36,38,96:GOTO1210
 1170 VDU23,230,60,60,24,24,28,22,19,50:
GOTO 1210
```




```
 1180 VDU23,230,60,60,24,24,56,232,72,24
:GOTO 1210
 1190 VDU23,230,60,60,24,24,56,104,200,7
6:GOTO 1210
 1200 VDU23,230,60,60,24,24,28,23,18,24:
GOTO 1210
 1210 ON M%+B%+1 GOTO 1220,1230,1240,128
0,1250,1260,1270,1280
 1220 VDU23,231,60,90,126,60,36,126,255,
189:GOTO 1280
 1230 VDU23,231,28,6,62,20,28,60,126,189
:GOTO 1280
 1240 VDU23,231,28,48,62,20,28,60,126,18
9:GOTO 1280
 1250 VDU23,231,189,219,255,189,165,126,
60,60:GOTO 1280
 1260 VDU23,231,157,135,191,149,157,126,
60,60:GOTO 1280
 1270 VDU23,231,157,177,191,149,157,126,
60,60:GOTO 1280
 1280 ENDPROC
 1290 :
 1300 DEF PROCmake
 1310 M2$=CHR$17+CHR$3+CHR$230+CHR$8+CHR
$11+CHR$17+CHR$1+CHR$231
 1320 B$=CHR$17+CHR$130+CHR$242+CHR$17+C
HR$128
 1330 M3$=M2$+CHR$8+CHR$11+B$
 1340 E1$=CHR$32
 1350 E2$=CHR$32+CHR$8+CHR$11+CHR$32
 1360 E3$=E2$+CHR$8+CHR$11+CHR$32
 1370 C$(0)=E1$
 1380 C$(1)=CHR$245+E1$
 1390 C$(2)=E1$+CHR$245+E1$
 1400 C$(3)=CHR$244+C$(2)
 1410 C$(4)=CHR$244+E1$+CHR$245
 1420 C$(5)=CHR$244+E1$
 1430 C$(6)=CHR$244
 1440 ENDPROC
 1450 :
 1460 DEF PROCman
 1470 IF IM%=0 THEN IM%=1 ELSE IM%=0
```

```
1480 IF Y%<26-F%*5 THEN F%=F%+1:IF 10*(
F%+1)-BL%>0 SC%=SC%+50*(F%+1)-5*BL%:COLO
UR1:PRINTTAB(13,3);SC%;
1490 K%=X%+1+((Y%+4)DIV5)*21+A%:M%=0:CO
LOUR1
1500 IF NOT(INKEY-104) OR V%<>0 THEN GO
TO 1530
1510 M%=1:H%=K%?-1:d%=?K%-H%:IF B%=4 AN
D H%<5 THEN B%=0:K%?-1=H%+1:PRINTTAB(X%-
1,Y%+d%);B$;TAB(X%,Y%-2);E1$;:SOUND1,1,1
00,6:GOTO 1560
1520 IF B%=0 AND H%>1 AND H%<6 AND (K%?
-21=0 OR ?K%<3) THEN B%=4:K%?-1=H%-1:PRI
NTTAB(X%-1,Y%+d%+1);E1$;:SOUND1,1,100,6:
GOTO 1560
1530 IF NOT(INKEY-105) OR V%<>0 THEN GO
TO 1560
1540 M%=2:H%=K%?+1:d%=?K%-H%:IF B%=4 AN
D H%<5 THEN B%=0:K%?1=H%+1:PRINTTAB(X%+1
,Y%+d%);B$;TAB(X%,Y%-2);E1$;:SOUND1,1,10
0,6:GOTO 1560
1550 IF B%=0 AND H%>1 AND H%<6 AND (K%?
-21=0 OR ?K%<3) THEN B%=4:K%?+1=H%-1:PRI
NTTAB(X%+1,Y%+d%+1);E1$;:SOUND1,1,100,6:
GOTO 1560
1560 IF INKEY-67 AND V%=0 THEN IX%=1:M%
=2
1570 IF INKEY-98 AND V%=0 THEN IX%=-1:M
%=1
1580 IF INKEY-99 AND V%=0 AND (B%=0 OR
(B%=4 AND K%?-21=0)) AND IX%=0 AND ?K%<3
THEN V%=-?K%:IY%=-1:SOUND1,-15,120,2:GO
TO 1690
1590 IF V%=0 THEN IY%=0:GOTO 1680 ELSE
IF V%>0 THEN V%=V%+1:IY%=1:M%=0:SOUND1,-
15,(7-V%)*20,4
1600 IF V%>6-?K% THEN V%=0
1610 IF V%=-1 THEN IY%=-1:V%=-2:SOUND1,
-15,150,2:GOTO1690
1620 IF V%=-3 AND TA%=1 THEN TA%=0:PRIN
TTAB(X%,Y%-3);B$;TAB(X%,Y%-3-K%?-21);E1$
;
1630 IF V%=-3 THEN V%=0:IF ?K%=1 THEN I
Y%=1
1640 IF V%=-2 AND B%=0 AND K%?-21>0 THE
N V%=-3:IY%=1:B%=4:K%?-21=(K%?-21)-1:SOU
ND1,1,100,5:TA%=1+((K%?-21)=0):GOTO 1680
1650 IF V%=-2 AND B%=4 THEN V%=-3:IY%=1
:B%=0:K%?-21=1:SOUND1,1,100,5:GOTO 1680
1660 IF V%=-2 THEN V%=-3:IY%=1:SOUND1,-
15,180,5
1670 IF V%=5 AND B%=4 THEN B%=0:K%?-21=
1
1680 IF B%=0 THEN M$=M2$:E$=E2$ ELSE M$
=M3$:E$=E3$
1690 PROCmanch:IF IY%=0 AND IX%=0 GOTO
1760
1700 IF K%?IX%=0 AND ?K%=1 AND Y%<26 AN
D K%?(IX%+21)<5 THEN V%=1:GOTO 1750
```




```
1710 IF ?K%=5 AND K%?(IX%-21)=1 THEN D%
=-1:GOTO 1750
1720 IF ?K%=1 AND Y%<26 AND K%?IX%=0 AN
D K%?(IX%+21)=5 THEN D%=1:GOTO 1750
1730 IF K%?(IX%-21)<>0 AND K%?IX%<>10 A
ND K%?IX%>3-B% DIV4 THEN IX%=0:GOTO 1750

1740 D%=?K%-K%?IX%:IF ABS(D%)>6 THEN D%
=0 ELSE IF ABS(D%)>1 IX%=0:D%=0
1750 PRINTTAB(X%,Y%);E$;
1760 Y%=Y%+IY%+D%:D%=0:IY%=0:X%=X%+IX%:
IX%=0:IF X%<0 THEN X%=19:Y%=Y%+?K%-K%?19
+(K%?19=0):V%=-(K%?19=0) ELSE IF X%>19 T
HEN X%=0:Y%=Y%+?K%-K%?-19+(K%?-19=0):V%=
-(K%?-19=0) ELSE GOTO 1780
1770 IF B%=4 THEN B%=0:M$=M2$:SOUND 1,1
,100,10
1780 PRINTTAB(X%,Y%);M$;
1790 ENDPROC
1800 :
1810 DEF PROCinst
1820 CLS:COLOUR3:PRINTTAB(5,2)"BUILDER
BOB";
1830 M%=0:IM%=0:B%=4:PROCmanch:PROCmake
:PRINTTAB(16,7)M3$;
1840 PRINTTAB(1,6)"Use the Bricks";
1850 PRINTTAB(0,9)"To build steps home"
;
1860 COLOUR2:PRINTTAB(6,12)"MOVE BOB";
1870 COLOUR1:PRINTTAB(0,15)"Z - LEFT  X
- RIGHT";
1880 COLOUR2:PRINTTAB(2,18)"PUT OR TAKE
BLOCK";
1890 COLOUR1:PRINTTAB(4,21)". - From LE
FT";TAB(4,24)"/ - From RIGHT";
1900 PRINTTAB(0,27)"SPACE - From ABOVE"
;
1910 COLOUR3:PRINTTAB(1,30)"RETURN TO C
ONTINUE";
1920 REPEAT UNTIL INKEY-74
1930 SC%=0:MEN%=3:ENVELOPE 1,1,6,0,-6,2
00,100,200,100,2,0,-1,120,110
1940 ENDPROC
```

```
 1950 :
 1960 DEF PROCscore
 1970 M%=0:B%=0:CLS:COLOUR3:PRINTTAB(4,2
)"SCORE SHEET";
 1980 IF MEN%<1 OR F%>4 THEN F%=0:COLOUR
3:PRINTTAB(4,6)"GAME OVER";:GOTO 2020
 1990 PRINTTAB(3,10)"MEN LEFT";
 2000 PRINTTAB(1,13)"RETURN TO CONTINUE"
;
 2010 PROCmanch:FOR I%=1 TO MEN%:PRINTTA
B(10+2*I%,10)M2$:NEXT:GOTO 2050
 2020 PRINTTAB(2,10)"FINAL SCORE ";SC%;:
IF SC%>HSC% THEN HSC%=SC%:COLOUR2:PRINTT
AB(2,14)"NEW ";
 2030 COLOUR2:PRINTTAB(6,14)"HIGH SCORE"
;:PRINTTAB(10,16);HSC%;
 2040 COLOUR1:PRINTTAB(4,20)"PRESS RETUR
N";:PRINTTAB(3,22)"FOR A NEW GAME";:GOTO
 2060
 2050 COLOUR2:PRINTTAB(2,6)"Score so far
 ";SC%;
 2060 BX%=1:BI%=1:M%=2
 2070 COLOUR1:COLOUR130:PRINTTAB(0,31);S
TRING$(20,CHR$242);TAB(0,26);STRING$(3,C
HR$242);TAB(6,26);STRING$(8,CHR$242);TAB
(17,26);STRING$(3,CHR$242);
 2080 FORI%=1TO4:PRINTTAB(3,26+I%);STRIN
G$(I%,CHR$242);TAB(17-I%,26+I%);STRING$(
I%,CHR$242);:NEXT:COLOUR128
 2090 OBY%=25:OBX%=0
 2100 REPEAT
 2110 BX%=BX%+BI%:IF BX%<0 THEN BX%=0:BI
%=1:M%=2 ELSE IF BX%>19 THEN BX%=19:BI%=
-1:M%=1
 2120 BY%=25:IF BX%>6 AND BX%<13 THEN BY
%=30:GOTO 2150
 2130 IF BX%>2 AND BX%<7 THEN BY%=23+BX%
:GOTO 2150
 2140 IF BX%>12 AND BX%<17 THEN BY%=42-B
X%
 2150 PRINTTAB(OBX%,OBY%)E2$;:PRINTTAB(B
X%,BY%)M2$;:OBY%=BY%:OBX%=BX%:FORW%=0TO3
00:NEXT
 2160 IF IM%=0 THEN IM%=1:IB%=1 ELSE IM%
=0:IB%=0
 2170 PROCmanch:UNTIL INKEY-74
 2180 ENDPROC
 2190 :
 2200 DEF PROCcrane
 2210 IF F%<1 THEN FOR W%=1TO100:NEXT:EN
DPROC
 2220 IF CP%>25 THEN CP%=CP%-1:ENDPROC
 2230 IF CP%>0 THEN CP%=CP%-1:GOTO 2250
ELSE CP%=30:VDU23,244,255,255,134,204,10
8,60,30,15:VDU23,245,255,255,97,51,54,60
,120,240
 2240 FL%=5-RND(6-F%):FL%=FL%-(FL%<F%):B
D%=0:DR%=RND(20)-1:P%=DR%+1+(6-FL%)*21+A
%:IF ?P%>3 THEN DR%=-6:ENDPROC ELSE BL%=
BL%+1:ENDPROC
 2250 IF CP%>22 THEN CR$=C$(CP%-19):GOTO
 2280
 2260 IF CP%<7 AND CP%>2 THEN CR$=C$(CP%
-3):GOTO 2280
 2270 CR$=C$(3)
 2280 IF CP%=DR%+5 THEN VDU23,244,255,25
5,134,204,120,56,24,12:VDU23,245,255,255
,97,51,30,28,24,48:BD%=5:PROCcheck2
 2290 IF ?K%>2-B% DIV4 THEN PROCcheck
 2300 IF CP%>5 COLOUR2:PRINTTAB(CP%-6,27
-FL%*5);CR$;:GOTO 2320
 2310 IF CP%>2 COLOUR2:PRINTTAB(0,27-FL%
*5);CR$;
 2320 IF CP%>4 AND CP%<25 AND CP%>DR%+5
THEN COLOUR1:PRINTTAB(CP%-5,27-FL%*5);B$
;
 2330 Q%=CP%-2+(6-FL%)*21+A%:IF CP%>2 AN
D CP%<23 AND ?Q%=5 THEN COLOUR1:PRINTTAB
(CP%-3,27-FL%*5);B$;
 2340 IF BD%>?P%+1 THEN PROCcheck2:BD%=B
D%-1:COLOUR1:PRINTTAB(DR%,31-FL%*5-BD%);
E1$;TAB(DR%,32-FL%*5-BD%);B$;:IF BD%=?P%
+1  ?P%=?P%+1
 2350 ENDPROC
 2360 :
 2370 DEF PROCcheck
 2380 IF X%=CP%-5 AND FL%=6-(Y%+4)DIV5 A
ND ER%=FALSE THEN ER%=TRUE:MEN%=MEN%-1:S
C%=SC% DIV2
 2390 ENDPROC
 2400 :
 2410 DEF PROCcheck2
 2420 IF X%=DR% AND ER%=FALSE AND ((FL%=
7-(Y%+4)DIV5 AND ?K%>3-B% DIV4) OR FL%=6
-(Y%+4)DIV5) THEN ER%=TRUE:MEN%=MEN%-1:S
C%=SC% DIV2
 2430 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS

WORDWISE PLUS AND ADDCOMM CLASH - S.Lazereno

The on-screen underlining and inverse video features of Wordwise Plus will not operate if Vine Micro's Addcomm ROM is installed in a higher priority ROM socket.

WORDWISE PLUS FUNCTION KEYS - C.B.Hill

From a Wordwise Plus menu, there is no need to use Shift along with a function key to access your own definitions, just the function key alone should be pressed.

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams
Assistant Editor: Geoff Bains
Production Editor: Phyllida Vanstone
Technical Assistant: Alan Webster
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft
Additional thanks are due to Sheridan Williams, Adrian Calcraft, John Yale and Tim Powys-Lybbe.

Editorial Address
BEEBUG
PO BOX 50
St. Albans
Herts.

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.

HINTS

There are prizes of £5 and £10 for the best hints each month, plus one of £15 for a hint or tip deemed to be exceptionally good.

Please send all editorial material to the editorial address below. If you require a reply it is essential to quote your membership number and enclose an SAE.

SUBSCRIPTIONS

Send all applications for membership, subscription renewals, subscription queries and orders for back issues to the subscriptions address.

MEMBERSHIP SUBSCRIPTION RATES

£ 6.40 6 months (5 issues) UK ONLY
£11.90 UK - 1 year (10 issues)
£18 Europe, £21 Middle East
£23 Americas & Africa, £25 Elsewhere

BACK ISSUES
(Members only)

Vol.	Single Issues	Volume sets (10 issues)
1	80p	£7
2	90p	£8
3	£1	£9
4	£1	–

Please add the cost of post and packing as shown:

DESTINATION	First issue	Each subsequent issue
UK	30p	10p
Europe	70p	20p
Elsewhere	£1.50	50p

All overseas items are sent airmail (please send a sterling cheque). We will accept official UK orders 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. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

Subscriptions, Back Issues &
Software Address

BEEBUG
PO BOX 109
High Wycombe
Bucks. HP10 8NP

Hotline for queries and software orders

St. Albans (0727) 60263
Manned Mon-Fri 9am-4.30pm

24hr Answerphone Service for Access and Barclaycard orders, and subscriptions
Penn (049481) 6666

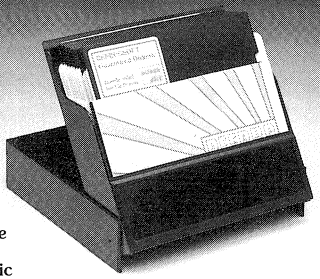
If you require members' discount on software it is essential to quote your membership number and claim the discount when ordering.

DYNAMIC DISCS

TESTED BY BEEBUG

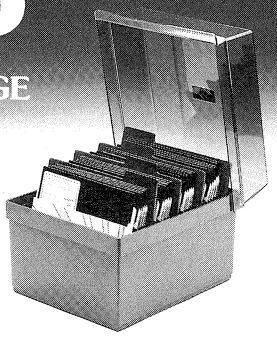
BEEBUG, the largest independent computer user group in the UK, offer 100% tested discs supplied by one of Britain's leading disc manufacturers.

10 FREE LIBRARY CASE



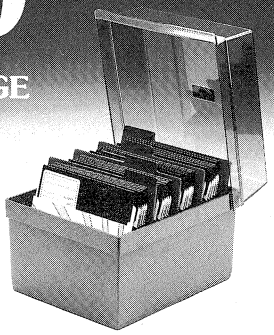
Orders for 10 discs are sent in black plastic library cases.

25 FREE STORAGE BOX



Orders for 25 are delivered in strong plastic Storage box with 4 dividers.

50 FREE STORAGE BOX



Orders for 50 are delivered in strong plastic Storage box with 4 dividers.

COMPARE PRICES

4 Types of Disc To Meet Your Exact Requirement

	48 TPI	DOUBLE DENSITY			
10	S/S D/D	£14.90	10	D/S D/D	£20.50
25	S/S D/D	£34.90	25	D/S D/D	£46.20
50	S/S D/D	£59.30	50	D/S D/D	£82.40

	96 TPI	DOUBLE DENSITY			
10	S/S D/D	£20.50	10	D/S D/D	£21.90
25	S/S D/D	£46.20	25	D/S D/D	£49.90
50	S/S D/D	£82.40	50	D/S D/D	£93.50

All prices include Storage Box, VAT and delivery to your door (UK)

Suitable for BBC Micro and all other computers using 5¼ inch discs including Atari and Commodore.

Fully Guaranteed — Not only by Beebug but by one of the UK's top disc manufacturers.

We regret that we have had to pass on a slight increase in the price of our discs, but we are now able to offer a wider range to meet your exact requirements.

These discs are the best. Please use the enclosed order form and order from our usual address. Beebugsoft, PO Box 109, High Wycombe, Bucks. HP10 8NP.

Official orders are welcome.

Barclaycard and Access telephone 0494 81 6666 Further information telephone 0727 60263

BEEBUG SOFT

Magazine Cassette/Disc

AUG/SEPT CASSETTE DISC CONTENTS

40/80 DUAL FORMAT UTILITY – provides 64K in both 40 and 80 track formats on the same disc

POP-UP CALCULATOR – instant calculations in any program including Wordwise

DRAWING SIMPLE BAR CHARTS – a useful program in our new series 'First Course'

ADDING NEW O.S. COMMANDS – with a complete demonstration of how it's done

WORKSHOP PROCEDURES – speeding up calculations with look-up tables

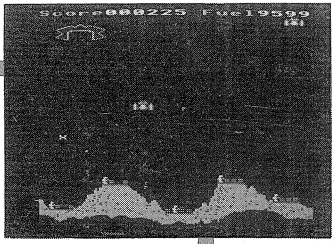
DISC CATALOGUE DISPLAY – an easily extended program to look in detail at your discs

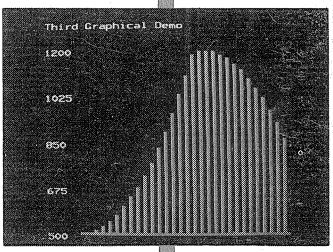
BUILDER BOB – challenging fast action for the games enthusiast

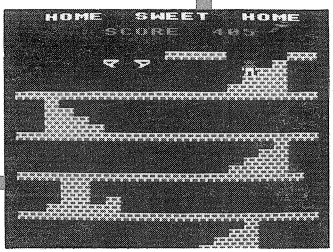
EXTRA FEATURE THIS MONTH

RESCUE MISSION – a very colourful and smooth version of this popular style game to rescue the human survivors in a hazardous mission

DISASSEMBLER – for second processor users this disassembles the full instruction set of the 65C02 processor and follows the Extended 65C02 Assembler published in BEEBUG Vol. 4 No. 1.







All this for £3.00 (cass) £4.75 (disc) +50p p&p.
Back issues (disc since Vol. 3 No. 1, cass since Vol. 1 No. 10) available at the same prices.

Subscription rates	DISC UK	CASS UK	DISC O'seas	CASS O'seas
6 months (5 issues)	£25	£17	£30	£20
12 months (10 issues)	£50	£33	£56	£39

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

Cassette subscriptions can be commuted to disc subscription on receipt of £1.70 per issue of the subscription left to run.

All subscription and individual orders to
BEEBUGSOFT, PO BOX 109, High Wycombe, Bucks, HP10 8NP.

Due to a hitch at the printers the main constructional diagram for this project, on page 21 of this issue, has unfortunately been printed incorrectly with essential information missing. The complete, and correct, version of this is reproduced in enlarged form below. Our thanks to the printers for speedily reprinting this diagram.