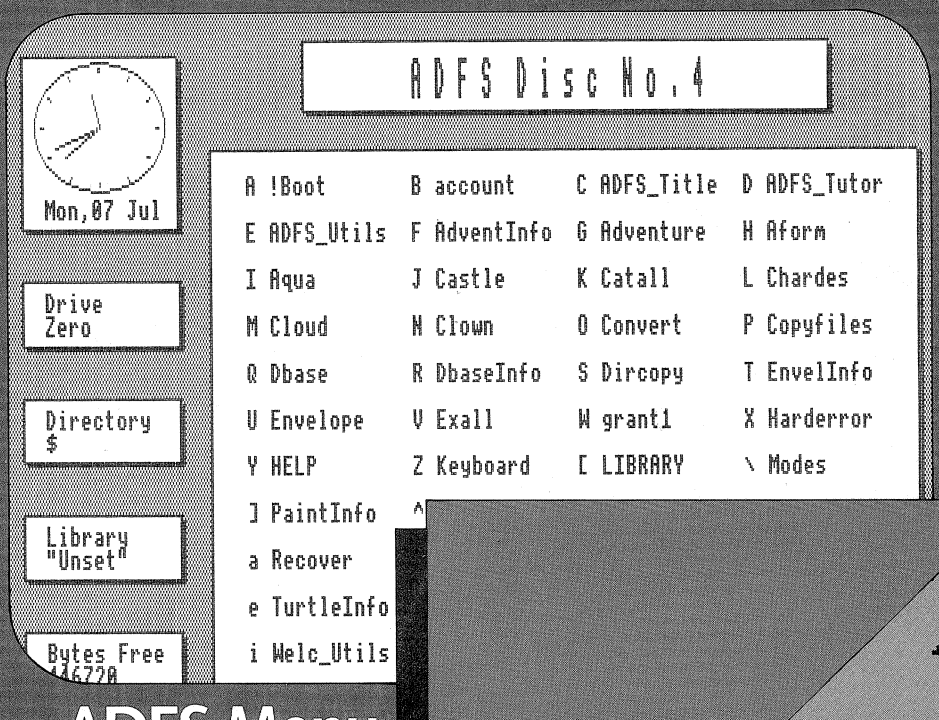
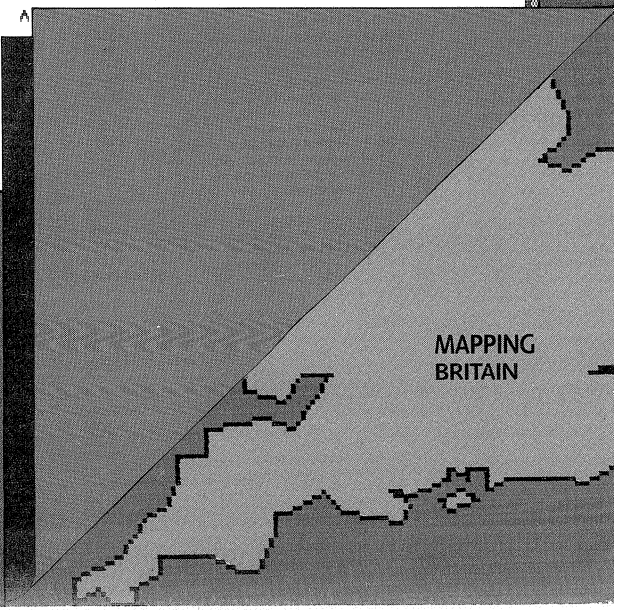
VOLUME 5 Number 4 AUG/SEPT 1986 PRICE £1.30

BEEBUG

FOR THE BBC MICRO



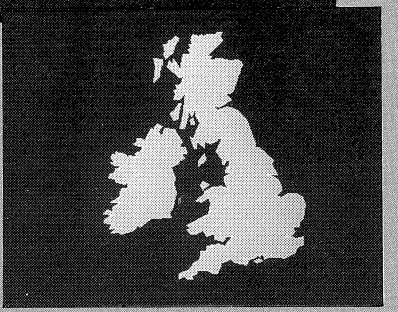
ADFS Menu



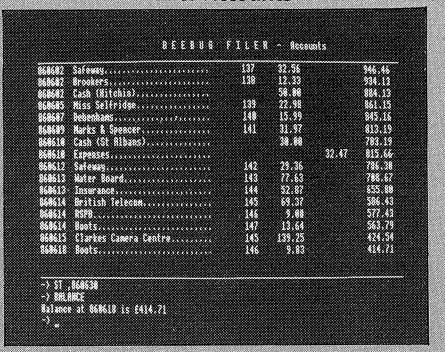
Mapping Britain



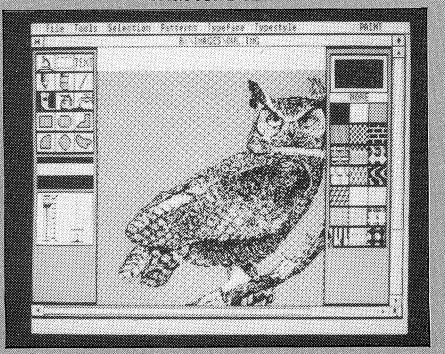
Painting by Numbers



Filer Accounts



Master 512



BEEBUG

VOLUME 5 NUMBER 4
AUGUST/SEPTEMBER 1986

FEATURES

Disc Manager 9
Mapping the British Isles 13
Painting by Numbers (Part 2) 14
Filer Accounts Option 19
Instant Texturing 26
Printer Graphics (Part 1) 40
Software for Sideways RAM (Part 3) 44
Posidroid 57

REVIEWS

The Master 512 6
Inter-word from Computer Concepts 16
Wordwise Plus Handbook 42
System Delta 46

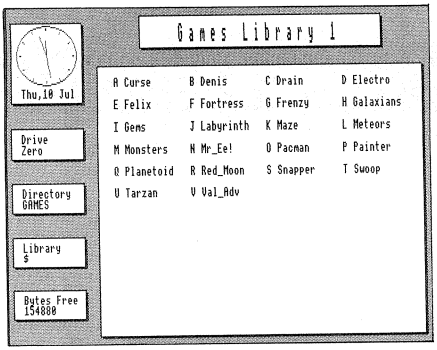
REGULAR ITEMS

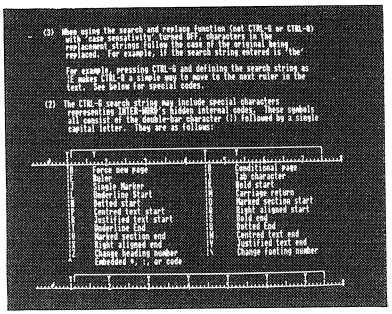
Editorial Jottings 3
News 4
BEEBUGSOFT Forum 5
The Master Series — An ADFS Menu 23
BEEBUG Workshop — Bitmaps 50
Postbag 52
Hints & Tips 53
First Course — Getting it Right 54
Points Arising 56

HINTS AND TIPS

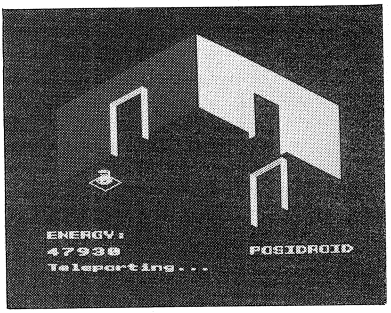
Determining Mode Legally
Wordwise Plus Line
Slow Writing
Star Command Parameters
Aries Boot
Cursor Position in Wordwise Plus
Software Tube Turn-off
FALSE, TRUE and True

ADFS Menu

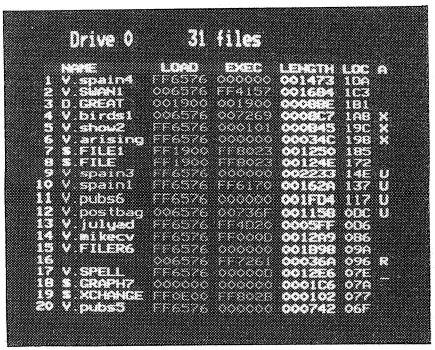




Inter-word



Posidroid



Disc Manager

EDITORIAL JOTTINGS

BEEBUG MAGAZINE

First of all, do remember that this issue of the magazine covers the two months of August and September. The next issue, which will be that for October, should be with you by the beginning of that month. This August/September issue seems to be particularly packed with programs, articles and reviews, and we have even used a few pages normally reserved for the supplement.

In this issue, the page numbering has been changed to include the supplement, and pages are now numbered continuously from beginning to end. We have tried various page numbering schemes for the supplement, but this seems to be the most helpful.

FREE PERSONAL ADS

As part of a move to add to the advantages of BEEBUG membership , we have decided to offer members the opportunity to advertise in the magazine completely free of charge. This is for personal (not business) ads only, and will enable you to sell any unwanted items of software and hardware. Please keep your ads as short as possible and send them in to arrive by the 12th of each month.

This will take effect with the October issue. We will endeavour to include all ads received but we do reserve the right to edit or reject those that we feel are unsuitable. Take advantage of BEEBUG and advertise to the many thousands of BEEBUG members. Send details of your free ad to PERSONAL ADS, BEEBUG, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX.

We will still continue to accept members' classified business ads at a charge of 30p (inc VAT) per word. Send details of your ad with a cheque for the required amount to the same address as above.

This is just the first of a number of new services that we are considering for BEEBUG members. We are also seeking ways in which we can respond more to the views and opinions of members. Apart from individual letters and phone calls, which are very useful in this respect, we do also carry out regular selective surveys of members, to keep us informed. In this way, we aim to improve further all aspects of BEEBUG, still Britain's largest independent computer user group and the only national user group catering for users of the BBC Micro and Master series.

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 for that configuration (normally some modifications will be required), and a cross through a symbol indicates a program that will not work on that type of system.

Basic I

Basic II

Tube

Model B+

Electron

Disc

Cassette

Master 128

News News News News News News Ne

Games by Phone

By the end of August Micronet promises to have a multi-user game up and running for all Micronet subscribers. Shades is the name of the game and it will cost players 99p an hour to have a go. Playing at off peak times will mean that calls are charged at local rates. Micronet is on 01-278 3143.

The much respected adventure game software house Level 9 is to about to launch a multi-user game too. Details are still to be decided but the game will feature over ten thousand locations, and a thousand computer controlled people to interact with. The game also has its own internal history and calendar running at twelve times 'real' time speed. Further details from Level 9 on 0734-595759.

Master Modem

Modem House - the manufacturer of the Voyager 7 modem - has now obtained BABT approval for that modem and released an internal modem for the Master.

The Voyager 7 modem costs £103. The Master 128 has provision for a modem to plug in inside the case. The internal modem is also going to cost £103. This isn't surprising as rumours have it that the internal modem is just a Voyager 7 circuit board with plugs to fit the Master. Despite this the BABT approval of the internal modem may take some time as this involves testing all possible combinations and versions of the computer as well as the modem. The internal modem will be supplied with Soft Machinery's Commsoft ROM - the same software as supplied with the BEEBUG Magic Modem. Further details from Modem House on 0392-213355.

User Ports for Users

WD Interfaces has released a range of equipment to help BBC micro users experimenting with building their own hardware. The Multiple User Port Adaptor provides from 4 to 32 extra User Ports derived from the 1MHz bus. The ports are fully buffered and plug compatible with the Beeb's own User Port. The base address of each pair can be set as desired. The Multiple User Port adaptors cost from £110.

If you are satisfied with a single User Port then the User Port Monitor may help you. This duplicates the User Port connector in a separate plastic case. An LED monitor light and 4mm jack plug is provided for each of the eight data lines too. The User Port Monitor costs £28. WD Interfaces is on (0532) 864328.

Let Your Micro Do the Walking

A new Electronic Yellow Pages telephone directory has been announced by British Telecom to come into service in January next year. The EYS will comprise a central database accessible by anyone with a suitable viewdata terminal and that, of course, will include BBC micro owners with a modem. Details from the system's designer, Intercom Data Systems on 04862-26951.

Master Mouse

The AMX mouse and software from AMS is now available for the Master 128. The Master version of the mouse comes with the AMX Super ROM on disc to download into the Master's sideways RAM. Pagemaker is also now available for the Master. Details from AMS on 0925-413501.

Acorn Reverses the Charges

Acorn Computers has revealed that early versions of the Master 128 have a problem with the battery backed RAM. The design of the Master may cause reverse charging of the battery at the end of the battery's life. This applies to Masters supplied by Acorn up to the end of May 1986.

A replacement battery package for all Masters affected is available free of charge through any Acorn dealer. Dealers will either dispatch the upgrade kit for fitting by users, or will fit the upgrade themselves if the Master is taken back to the dealer. Acorn is on 0223-214411.

BEEBUG SOFT FORUM

Masterfile and the Master

SETTING DRIVE ZERO

Because Masterfile is written in Basic, and uses fully 'legal' code, it is directly transferable to the Master. There is just one single modification required on certain versions of Masterfile, and this concerns the setting of the default drive in the initialisation routine. To do this, place your program disc in the drive, and proceed as follows:

```
*ACCESS LF <Return>
LOAD "LF" <Return>
```

Now find the line that contains *DRIVE 0 and delete any other text, e.g. REM statements, on the same line. Make sure that the line ONLY contains *DRIVE0.

```
SAVE "LF" <Return>
*ACCESS LF L <Return>
```

AUTOMATIC DATE READING

Since the Master knows not only the exact time of day, but also the day, month and year, it seems an annoying duplication of effort that Masterfile Master users should have to tell the machine what it already knows each time Masterfile is booted up.

The short function listed later will read the Master's clock/calendar and put the result in a format compatible with all versions of Masterfile (i.e. dd.mm.yy). In fact, all but the earliest versions of Masterfile I allow a free format date input. You can then use the function to set the Masterfile date variable ($&C00) accordingly. There is also a statement to print the date to the screen. The function will work just as well in your own programs, provided that you are already in mode 7 when it is called.

You will see that the date is read in from the calendar chip using TIME$ rather than OSBYTE calls. A small snag with this is that the month is read in string format (i.e. "Jul" instead of "07"). The new routine translates this in lines 2030 to 2040 using data contained in line 2050. Any necessary adjustment, by padding out months 1-9 with a leading zero to keep to the dd.mm.yy format, is performed in line 2040.

If the routine is run on a standard model B, it will throw up a "Type mismatch" error. If you are likely to be using Masterfile on both BBC Bs and Masters, you may wish to add the following line which exits from the function if a Master is present:

```
2015 IF INKEY(-256)<>253
     THEN =""
```

To enter the routine into Masterfile, you should proceed as follows:

```
Insert program disc
*ACCESS LF
LOAD "LF"
```

Type in the routine. Now locate the line that contains $&C00="" and replace this line with $&C00=FNsetdate. Should your program not contain such a line, simply insert it anywhere at the beginning of the program. Now save the modifications with:

```
SAVE "LF"
*ACCESS "LF" L
```

Test the routine and the job is done. But please note that when you are testing the combined routine, your new routine will be overwritten when program LF chains in LFmain. To avoid this, you could ensure that the disc drive has no disc in it when you test-run the program. Then press Escape to exit the attempted load.

```
2000 DEF FNsetdate
2010 REM Set date on Master
2020 month$=MID$(TIME$,8,3)
2030 month=0:REPEAT:month=
month+1:READ name$:UNTIL na
me$=month$
2040 month$=STR$(month):IF
LEN(month$)=1 month$="0"+mo
nth$
2050 DATAJan,Feb,Mar,Apr,Ma
y,Jun,Jul,Aug,Sep,Oct,Nov,D
ec
2060=MID$(TIME$,5,2)+"."+mo
nth$+"."+MID$(TIME$,14,2)
```

The line:

```
IF INKEY(-256)=253 THEN
PRINT TAB(7,22) CHR$(134);
"Today's Date:";CHR$(131);
$&C00:A=INKEY(100)
```

will display the newly created date, and pause for one second so that you can admire it. Should you wish to incorporate this line, simply insert it immediately before the line containing:

```
program_name$=".....
```

ADFS VERSION

We expect to be bringing out an ADFS version of Masterfile shortly. Watch this space.

THE MASTER 512

The 512 upgrade for Acorn's Master Series at last offers the potential of a real business system from Acorn. Peter Rochford has been putting the new machine through its paces.

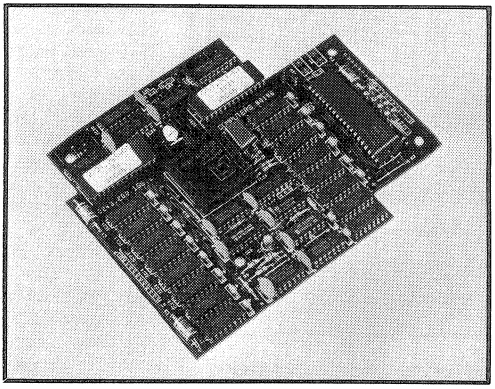
Product : Master 512 Upgrade
Supplier : Acorn Computers Ltd
: Cambridge Technopark,
: 645 Newmarket Road,
: Cambridge CB5 8PD.
: Tel. 0223-214411
Price : £399

In the world of business computing, very few machines these days operate with an 8 bit microprocessor. The new generation of true 16 bit chips has far greater processing speed and addresses larger amounts of RAM, allowing much more powerful software to be run. The market is dominated by IBM with its PC (although not true 16 bit) and variants, along with the countless IBM clones and compatibles from other manufacturers. All these machines and many others use MS-DOS in one version or another, and this has virtually become the standard operating system on small business computers.

THE MASTER 512

Acorn have now released their first 16 bit machine, the Master 512 that comprises a Master 128, with a co-processor running an 80186 microprocessor at 10MHz and with 512k of RAM. Existing Master owners can upgrade their machine by buying the 512 board and plugging it into the main PCB, as with the 8 bit Turbo board, reviewed in BEEBUG Vol.5 No.2.

The Master 512 uses an operating system called DOS Plus from Digital Research that supports IBM PC-DOS, MS-DOS and CPM/86 applications. Supplied with the Master 512 are two manuals, a mouse, three system discs containing the bundled GEM software and the bootable operating system disc.



BOOTING THE SYSTEM

Activating the 512 co-processor is quite straightforward. The bootable DOS Plus disc which is in ADFS format, is placed in drive 0. After a *CONFIGURE TUBE, you press Ctrl-Break and the screen displays the Digital Research name and logo. At this point you leave the Master OS and Basic as the bootstrap ROM in the co-processor takes over, loading the DOS Plus operating system from disc into the 512.

DOS PLUS

It is not within the scope of this review to explain fully the workings of DOS Plus as it is a powerful and sophisticated operating system. However, users of the Acorn ADFS will feel very much at home with its file handling because of its similar hierarchical structure. Many commands are similar, and adapting to it presents little problem - apart from the constant desire to put a 'star' in front of every command which it does not need! The Master's on-board clock is used by DOS to automatically date and time stamp files as they are created or updated.

As with the ADFS, both sides of a disc are used and two floppies are supported as drive A and B, plus a hard disc, drive C. Although not mentioned in the manual, a RAM disc can be used as part of the filing system via a utility called MEMDISK and this is designated as drive M.

Earlier, I said that once DOS is loaded you leave the Master OS. This is not strictly true, as the 512 still uses the

Master I/O processor to take care of the screen, ports and keyboard etc. DOS Plus for the Master 512 is an extended version, modified especially for Acorn, and communicates with the Master OS. To the user all this is transparent, but you can still communicate directly with the Master OS if required via a STAR utility. What you can do has limitations obviously, but you can use FX calls (with caution!) and things like *HELP and so on.

THE BOOT DISC AND UTILITIES

The bootable DOS disc contains a large number of files apart from DOS Plus itself. These are utilities for use with DOS and many are unsupported by the documentation. There is however a large on-screen help facility on the disc that gives brief details and examples of their uses.

The DISK file is a formatter, verifier and copying application. This enables the backing up and formatting of discs in a number of different formats. These are listed in Fig. 1. The ability to read and write to discs in a large number of DOS and CPM formats is a very important and impressive feature of the the Master 512. You will notice from the list that Acorn have their own DOS format giving 800K on one disc using both sides.

Those who are interested in transferring their existing text files to and from the 512 will be pleased to know that this is possible via a special utility. In fact, this review was written with 'GEM Write' on the 512, and the file transferred to ADFS format and read into View before transferring via modem direct to BEEBUG.

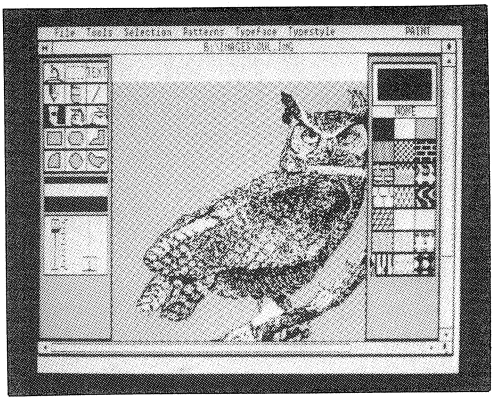
Leaving the 512 and returning to the Master 128, is achieved by calling a file on the utility disc called NOTUBE which cleanly exits and places you back with the I/O processor and Basic. This file needs to be transferred to your other working discs but is not entirely necessary as you can press Break and then at the star prompt, enter *CONFIGURE NOTUBE and do a Ctrl-Break.

640K DOS Bootable DOS disc
800K DOS ACORN DOS Format
360K DOS IBM 512 x 9
720K DOS TANDY 512 x 9
720K DOS OLIVETTI 512 x 9
720K DOS NIMBUS 512 x 9
400K CPM ACORN Z80 Format
320K CPM IBM 512 x 8
720K CPM ALTOS 512 x 9

Fig.1

Further utilities of note on the boot disc are those for COLOUR and ALARM. COLOUR lets you change the background and foreground colours of the screen to any supported by the Master. ALARM is most intriguing. As it suggests, it enables alarms to be set up and run as background tasks whilst you get on with something else. A number of these may be set and when activated produce a rather unmusical melody along with the message previously entered displayed on the screen.

Under the heading of utilities I should make mention of the fourth disc supplied with the 512. This contains several other undocumented utilities along with a SETUP application for configuring GEM. Amongst the utilities is a Z80 emulator program. When loaded in, this enables CP/M-80 applications to run, albeit at a clock rate of only 1.5Mhz.



THE GEM COLLECTION

Two of the discs supplied with the Master 512 contain the GEM bundled software. GEM stands for Graphics Environment Manager and it is, to use a common buzzword, a WIMP system (Windows, Icons, Mouse, Pointer). I imagine most Beeb owners are now familiar with this style through the AMX software or possibly that of the Apple Mac.

The mouse supplied with the 512 plugs into the user port. The one I had for this

review is not the one that will be supplied with the final package, I am pleased to say, and in fact, I used my own Wigmore Megamouse which worked perfectly, although GEM only requires a two-button mouse.

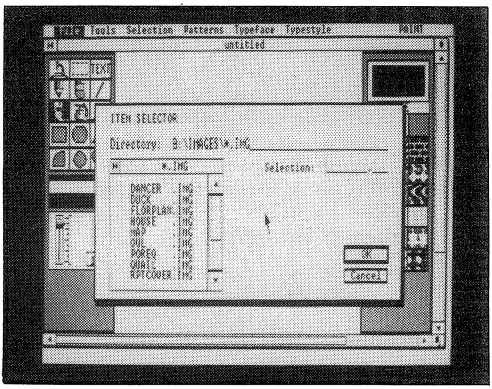
The GEM Collection comprises GEM Desk Top, GEM Paint and GEM Write. Taking GEM Desk Top first, this is a file management system using the mouse and drop-down menus. This enables the selection and manipulation of files from discs without resorting to the use of DOS at system level. It is delightfully simple to use. Included amongst its many facilities are a clock/calendar with alarm, a calculator that has memory functions, a 'Snapshot' facility for saving screens and a print spooler. Like the rest of GEM, GEM Desktop is in 80 column with two colours. You can reconfigure the system to give 4 colours and 80 column for all parts of GEM, but this is a 'fudged' Master mode 1 screen and the definition is really rather poor.

GEM Paint is an artwork or image creation application similar to things like MAC Paint, AMX ART, Pagemaker etc. There are the usual drawing facilities, patterns, fills, various brushes and so on. There is a choice of fonts for text and a large selection of patterns that can be loaded in from disc. Like Desk Top, there is also the facility to save screens via 'Snapshot', and load in others saved from any section of GEM.

GEM Write is the word-processor part of GEM. This, like GEM Paint, is selected from GEM Desk Top. If you use View or Wordwise then you know already the kind of thing to expect. It has very similar editing facilities with the usual Move, Copy, Search and Replace. It operates in 80 columns and can be controlled via mouse, function keys, or, Ctrl and letter keys, the latter being identical to those used in Wordstar.

There are certain features in GEM Write that you won't find in View or Wordwise. italic and bold are shown on screen, as are paragraph markers too. More impressive though, is the ability to insert images created in GEM Paint into the text.

Amazingly for a system with 512k of RAM, text in GEM Write cannot be printed from memory, but is printed via the



'Output' application provided, straight from disc. Another surprise is that the amount of text that can be held in memory amounts to only a few pages, the rest being held in 'spill' files on disc. This is no fault of the Master 512 but rather that of GEM which gobbles up a massive amount of RAM.

IBM COMPATIBILITY

Acorn do not claim that the Master 512 is an IBM compatible or indeed a clone. It will run applications for the IBM however, but with limitations. Emulating the screen modes of the IBM is done with a utility on the boot disc. This will provide all the modes available on the IBM Colour Graphics Adaptor but modes 0 to 3 which allow eight colours on the IBM have only two colours on the 512. Another thing missing is the lack of an extra brightness level for emphasizing. Acorn have got round this by the inclusion of an extra bold-faced character set, although I must say that the legibility is not that good.

I did, during the course of this review, lay hands on several IBM software packages to run on the 512. Most interesting, and probably the most popular around, is Lotus 123 which ran perfectly, being a Release 1 version. Release 2 will not work because of the disc protection built into the software. Acorn tell me that disc protection and hardware dependent features are the main limiting factors for certain IBM software not running on the Master 512. They do claim however, that 60-70% of IBM software will run, and they intend to publish a list which will be available shortly from Acorn dealers.

→28

DISC MANAGER

If you find looking after files on disc Bernard Hill's Disc Manager will take laborious and time consuming, care of all your needs.

When using a disc system, how often have you wished that the *CAT or *INFO*.* commands were more versatile? The first is too brief for detailed use, the second scrolls past and you've lost the details you wanted. In any case, neither orders the files in any really sensible way. If you're like me you have a disc with various files on, the names give no indication of function, and even when you've found out their function, deleting or renaming is a tedious chore of unlocking, rechecking, deleting...

The program on which this article is based arises directly from my own frustrations. It enables many file operations to be specified and checked and then carried out with a single command.

On running the program, the display clears to a mode 7 screen with information about the drive specified, the number of files, and, if they have been specified, the auto-boot option and disc title. Up to 20 files are then exhibited on the screen. Locked file names appear in green and all files have their length, load, execution and disc addresses shown. The cursor has moved to the first file under a column labelled A (for 'Action'). This cursor can be moved among the files with the cursor keys, either one at a time or 20 (a screenful) at a time. Now a single press of a key brings a choice of actions of two basic types: immediate actions, and actions on individual files which are delayed until the "go" command is given.

THE IMMEDIATE ACTIONS

Esc: Stop the program
H: Display a help screen.
f0-f3: Re-catalogue with drive 0-3.
N: Re-sort the files into alphabetic order, by directory plus name.
O: Re-sort the files into the order on which they occur on the disc.
-: Re-sort the files into descending order of size.
+: Ditto, but ascending order.
Ctrl-@: Compact the disc, and re-run the program. For this to work the program must be saved on the disc under the name "DISCMAN"
S: Show a disc "map". For 80-track discs mode 3 is selected, mode 6 for 40 tracks. Each track is shown horizontally, and the 10 sectors on each are vertical. The first two sectors are marked "C" as they are the catalogue area. The file opposite the cursor is shown in white, other files in grey. Files marked for deletion (see below) are shown as empty boxes and files marked to be transferred to tape are shown with a cassette symbol. This options enables you to see the fragmentation of the disc and so decide whether it is necessary to compact the disc.
Ctrl-G: Go (execute the delayed actions - see below)

The program may be used as a menu as the following commands are available:
Ctrl-C: CHAIN the file opposite the cursor.
Ctrl-R: *RUN the file.
Ctrl-E: *EXEC the file.

THE DELAYED ACTIONS

This group of operations is on specific files and is not executed until Ctrl-G is pressed. The "A" column on the screen shows the action intended.

X: Mark for deletion. Note that this operation acts on locked files also, unlike the standard DFS command.
0-3: Mark for *COPYing to another disc
T: Mark for backup to tape, allowing an unattended, and selective, bulk back-up from disc to tape. Note that the filename on tape will also contain the directory: e.g. "$.FILE". All file transfer operations can be on files of any size.
L,U: Mark for locking or unlocking
?: Mark for *DUMPing. Very useful when you've forgotten what a file was!
R: Mark for rename. On a press of R the cursor moves to the beginning of the file name (which is now red) and allows the name to be overtyped with the new name. Note that the name is replaced character by

character and not completed until <Return> is pressed, at which point all the old characters not changed are reinstated, thus allowing easy changing of the directory alone. Characters can be erased by replacing them with spaces. Just as with deletions, the lock status of the file is ignored.

Now press Ctrl-G and watch it happen! First the transfers to tape take place. Since this can be a lengthy operation some guide is given to progress in the form of a disc map with sectors ticked off as they are completed. Afterwards come all the other operations, the order of actions being that of the last-displayed screen.

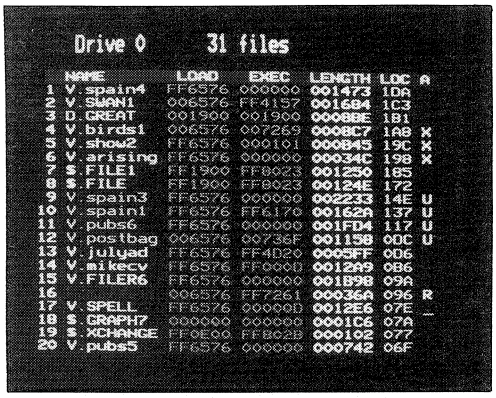
PROGRAM CUSTOMISING

Since the program is large it uses part of the DFS workspace to store some arrays. If you are using it on the Master, replace line 220 with:

```
220 DIM sec 799, page 255,new 255
```

It is very useful to have this program residing in ROM. Before you use ROMIT (or equivalent), to make a RFS image, replace line 2710 with:

```
2710 *K.0X%=&80:Y%=&3:CALL&FFF7|M
     *ROM|MCH."DISCMAN"|M
```



ERROR REPORTING

It is possible that the program may stop with DFS errors (e.g. *COPY to a disc which contains a locked file of the same name). In this case the program will stop with the usual error message. To reset everything, press Break (or re-run the program). The communication between subsequent runs of the program is via the resident integer variable D%. If this has the value 0 to 3 then this is taken as the number of the drive to be catalogued. If you have no formatted disc in any of these drives then the program will hang or give a drive error. Set D% and re-run. If you have a single drive you may like to replace line 290 with D%=0.

The program conforms to the normal Acorn DFS single density format. A separate version to cope with the Watford DFS 62 file catalogue is also included on this month's magazine cassette/disc.

```
  10 REM PROGRAM Disc Manager
  20 REM VERSION B0.3
  30 REM AUTHOR  Bernard Hill
  40 REM BEEBUG  AUG/SEPT 1986
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 MODE7:HIMEM=&4000:m=31:*DISC
 110 *FX4,1
 120 *FX180,64
 130 *FX225,150
 140 *FX229,1
 150 *FX21
 160 VDU23,255,&7F7F;&7F7F;&7F7F;&7F7F;
 170 VDU23,254,&1515;&1515;&1515;&1515;
 180 VDU23,253,&637F;&6363;&6363;&7F63;
 190 VDU23,252,&677F;&7E66;&667E;&7F67;
 200 VDU23,251,3,6,12,&D8,&70,32,0;
 210 DATA "",(Load),(Run),(Exec)
 220 sec=&1300:page=&1200:new=&1400
 230 DIM lock m,cmd m,P m
 240 DIM code 54,p$(1)
 250 DIM name$(m),len%(m),lo%(m),ex%(m)
 260 DIM beg%(m),boot$(3):RESTORE 210
 270 FOR i=0 TO 3:READ boot$(i):NEXT
 280 PROCasmbl:p=page+8:p$(0)="s"
 290 IF D%<0 OR D%>3 THEN D%=0
 300 PROCrdsec(D%,0,1,page):p$(1)=" "
 310 nf=page?5 DIV 8:D$=CHR$(48+D%)
 320 page?4=0:T$=FNstr(page)
 330 boot=(page?6 DIV 16) AND 3
 340 ns=256*(page?6 AND 3)+page?7
 350 IF ns>400 THEN mode=3 ELSE mode=6
 360 FOR i=1 TO nf:lo%(i)=FNw(p,p+6,4)
 370 ex%(i)=FNw(p+2,p+6,64)
 380 len%(i)=(p!4 AND &FFFF) + 65536*((
p?6 DIV 16) AND 3)
 390 beg%(i)=p?7+256*(p?6 AND 3):p=p+8
 400 NEXT
 410 PROCrdsec(D%,0,0,page):p=page+8
 420 FOR i=1 TO nf:P?i=i:cmd?i=32
 430 b$=FNuc(CHR$(p?7 AND &7F))+"."
 440 lock?i=p?7 DIV 128:p?7=13
```

```
  450 name$(i)=b$+$p:p=p+8:NEXT
  460 page?8=0:T$=FNstr(page)+T$
  470 PROChead:PROCsort("P?")
  480 f=1:REPEAT PROCdisplay(f):c=GET
  490 IF c>96 AND c<123 THEN c=c-32
  500 IF FNcmd(c) THEN cmd?(P?f)=c
  510 IF c=0 THEN PROCcompact
  520 IF c=3 THEN PROCstop(P?f)
  530 IF c=5 PROCexec("EXEC",name$(P?f))
  540 IF c=7 THEN PROCgo
  550 IF c=18 PROCexec("RUN",name$(P?f))
  560 IF c=27 THEN PROCstop(0)
  570 IF c=138 THEN f=FNmin(f+1,nf)
  580 IF c=139 THEN f=FNmax(f-1,1)
  590 IF c=137 THEN f=FNmin(f+20,nf)
  600 IF c=136 THEN f=FNmax(f-20,1)
  610 IF c=45 THEN PROCsort("-len$")
  620 IF c=43 THEN PROCsort("len$")
  630 IF c=79 THEN PROCsort("beg$")
  640 IF c=78 THEN PROCsort("name$")
  650 IF c=82 THEN PROCnewname(f)
  660 IF c=83 THEN PROCusage(P?f,TRUE):V
DU22,7:PROChead
  670 IF c=72 THEN PROChelp
  680 IF c>149 THEN D%=c-150:RUN
  690 UNTIL FALSE
  700 :
 1000 DEFPROChelp:CLS:RESTORE 1050
 1010 c$=STRING$(21," ")
 1020 REPEAT READ a$,b$,c$
 1030 PRINTa$,TAB(10);b$;TAB(18);c$
 1040 UNTILb$="R":oldf=-20:PRINT"Any key
 to return...";:x=GET:ENDPROC
 1050 DATA Actions:,f0-f3,Change drive 0
-3
 1060 DATA ,^ v,Move 1 file
 1070 DATA ,[ ],Move 1 page
 1080 DATA ,N,Sort name order
 1090 DATA ,O,Sort disc order
 1100 DATA ,-,Sort descending size
 1110 DATA ,+,Sort ascending size
 1120 DATA ,S,Show file on disc
 1130 DATA ,Ctrl-@,Compact disc
 1140 DATA ,Ctrl-G,Execute file commands
 1150 DATA ,Ctrl-R,*RUN the file
 1160 DATA ,Ctrl-E,*EXEC the file
 1170 DATA ,Ctrl-C,CHAIN the file
 1180 DATA ,Escape,Stop,Files:,X,Delete
 1190 DATA ,"0-3,T","Copy to disc 0-3,Ta
pe"
 1200 DATA ,"L,U","Lock,Unlock",,?,Dump
 1210 DATA ,R,Rename
 1220 :
 1230 DEFPROCrdsec(d,t,s,a)
 1240 ?&70=d:!&71=a:?&79=&21
 1250 !&75=&10000*(256*s+t)+&5303
 1260 A%=&7F:X%=&70:Y%=0:CALL-15:ENDPROC
 1270 :
 1280 DEFFNw(x,y,n):z=!x AND &FFFF
 1290 IF ((?y DIV n) AND 3)=0 THEN =z EL
SE =z+&FFFF0000
 1300 :
 1310 DEFFNstr(addr):LOCAL i,a$:i=-1
 1320 REPEAT i=i+1
 1330 IF addr?i>0 THEN a$=a$+CHR$addr?i
 1340 UNTIL addr?i=0:=a$
 1350 :
 1360 DEFFNcmd(n)
 1370 RESTORE 1390:REPEAT READ ok
 1380 UNTIL n=ok OR ok=-1:=n=ok
 1390 DATA 32,48,49,50,51,63,76,84,85,82
 1400 DATA 88,-1
 1410 :
 1420 DEFPROCsort(a$):LOCAL P%,I%,J%,T%,
F%:CLS:P%=nf-1:IF P%=0:ENDPROC
 1430 VDU31,15,8,136:PRINT"Sorting....";
 1440 REPEAT:F%=TRUE:FOR I%=1 TO P%
 1450 IF FNa(I%)>FNa(I%+1) THEN T%=P?I%:
P?I%=P?(I%+1):P?(I%+1)=T%:F%=FALSE
 1460 NEXT:P%=P%-1:UNTIL F%:f=1:oldf=-20
 1470 ENDPROC
 1480 :
 1490 DEFFNa(x)=EVAL(a$+"(P?("+STR$x+"))
")
 1500 :
```

```
Drive 0      31 files
   NAME          LOAD   EXEC  LENGTH LOC A
Actions:  f0-f3    Change drive 0-3
          ^ v      Move 1 file
          [ ]      Move 1 page
          N        Sort name order
          O        Sort disc order
          -        Sort descending size
          +        Sort ascending size
          S        Show file on disc
          Ctrl-@   Compact disc
          Ctrl-G   Execute file commands
          Ctrl-R   *RUN the file
          Ctrl-E   *EXEC the file
          Ctrl-C   CHAIN the file
          Escape   Stop
Files:    X        Delete
          0-3,T    Copy to disc 0-3,Tape
          L,U      Lock,Unlock
          ?        Dump
          R        Rename
Any key to return..._
```

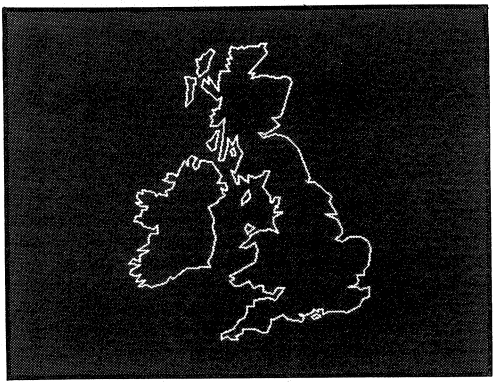
```
 1510 DEFPROChead:VDU26,30:CLS:@%=2
 1520 s$=CHR$132+CHR$157+CHR$135+CHR$141
+"Drive "+D$+"      "+STR$nf+" file"+p$(
-(nf=1))+"      "+boot$(boot)
 1530 PRINTs$'s$'LEFT$(s$,3);SPC7;
 1540 IF T$<>"" THEN PRINT"Title : "+T$
ELSE PRINT
 1550 PRINTCHR$129;CHR$157;CHR$135"NAME"
;TAB(14);"LOAD";TAB(21);"EXEC";TAB(27);"
LENGTH";TAB(34);"LOC A"
 1560 VDU28,0,23,39,4:oldf=-20:ENDPROC
 1570 :
```

```
1580 DEFPROCdisplay(n):LOCAL i,p
1590 IF nf=0 THEN PRINTTAB(10)"No files
":ENDPROC
1600 IF (n-1) DIV 20=(oldf-1) DIV 20 TH
EN 1720
1610 CLS:FOR i=(n-1)DIV20*20+1 TO FNmin
((n-1)DIV20*20+20,nf):p=P?i
1620 IF lock?p THEN a$=CHR$130 ELSE a$=
CHR$131
1630 @%=2:PRINTi;
1640 IF cmd?(P?i)<>82 THEN PRINTa$;name
$(p); ELSE VDU129:FOR r=0 TO 8:VDUnew?(9
*P?i+r):NEXT
1650 PROChex(133,lo%(p) AND &FFFFFF,6)
1660 PROChex(32,ex%(p) AND &FFFFFF,6)
1670 PROChex(135,len%(p),6)
1680 PROChex(134,beg%(p),3)
1690 VDU135,cmd?p
1700 IF i MOD 20<>0 THEN PRINT
1710 NEXT:oldf=n
1720 VDU31,38,(n-1) MOD 20,cmd?(P?n),8
1730 ENDPROC
1740 :
1750 DEFPROCstop(n)
1760 *FX4
1770 *FX225,1
1780 *FX229
1790 VDU26,30,12:PROChwm
1800 IF n>0 THEN CHAIN(":"+D$+"."+name$
(n))
1810 END
1820 :
1830 DEFFNuc(s$)
1840 IF s$>="a" AND s$<="z" THEN =CHR$(
ASCs$-32) ELSE =s$
1850 :
1860 DEFFNmax(a,b):IF a>b THEN=a ELSE=b
1870 DEFFNmin(a,b):IF a<b THEN=a ELSE=b
1880
1890 DEFPROCusage(n,p)
1900 LOCAL i,s,u,t:!sec=&20204343
1910 VDU22,mode,23;11,0;0;0;0
1920 FOR s=4 TO 796 STEP 4
1930 sec!s=&20202020:NEXT:u=2
1940 IF nf=0 THEN 2030
1950 FOR i=1 TO nf
1960 end=beg%(i) + len%(i) DIV 256
1970 IF len%(i) MOD 256=0 AND len%(i)<>
0 THEN end=end-1
1980 FOR s=beg%(i) TO end
1990 sec?s=255+(i<>n)
2000 IF cmd?i=88 THEN sec?s=253
2010 IF cmd?i=84 THEN sec?s=252
2020 u=u+1:NEXT:NEXT:@%=0
2030 PRINT'"Drive ";D$;SPC4;nf;" files
 Title:";T$'"Filename :"name$(n)'"Track
s ------>":@%=10:VDU32
2040 IF mode=3 THEN PRINT1,2,3,4,5,6,7
ELSE PRINT1,2,3
2050 PROCnum
2060 FOR s=0 TO 9:FOR t=0 TO ns DIV10-1
2070 VDU sec?(10*t+s):NEXT:NEXT
2080 PROCnum:@%=&20104
2090 PRINT'"Disc usage: "u/ns*100;"%"
2100 IF NOT p THEN 2130
2110 PRINT'"Spacebar to return....";
2120 REPEAT x=GET:UNTIL x=32
2130 VDU23;11,255;0;0;0;0:ENDPROC
2140 :
2150 DEFPROCnum:LOCAL i
2160 FOR i=0 TO ns DIV 10-1:VDU48+i MOD
10 : NEXT:ENDPROC
2170 :
2180 DEFPROCgo
2190 tn=0:CLS:FOR i=1 TO nf:p=P?i
2200 IF cmd?p=84 THEN PROCtape(p)
2210 NEXT:VDU22,7:FOR i=1 TO nf:p=P?i
2220 IF cmd?p>47 AND cmd?p<52 THEN PROC
copy(p)
2230 IF cmd?p=76 THEN PROClock(p," L")
2240 IF cmd?p=85 THEN PROClock(p,"")
2250 IF cmd?p=82 THEN L=lock?p:PROClock
(p,""):PROCrename(p):IF L=1 THEN PROCloc
k(p," L")
2260 IF cmd?p=88 THEN PROClock(p,""):PR
OCdelete(p)
2270 IF cmd?p=63 THEN PROCdump(p)
2280 NEXT:RUN
2290 :
2300 DEFPROCtape(n):LOCAL bk,t,s,L,b,by
2310 CLS:*TAPE
2320 *OPT1,0
2330 IF tn>0 THEN 2370
2340 *MOTOR1
2350 PRINT "RECORD then RETURN"
2360 REPEAT s=GET:UNTIL s=13
2370 tn=tn+1:t=OPENOUTname$(n)
2380 !&38C=lo%(n):!&390=ex%(n)
2390 IF len%(n)=0 THEN 2490
2400 s=beg%(n):L=len%(n) DIV 256
2410 PROCusage(n,FALSE)
2420 VDU23;11,0;0;0;0:FOR bk=0 TO L
2430 PROCrdsec(D%,s DIV 10,s MOD 10,pag
e)
2440 IF bk<>L THEN by=256 ELSE by=len%(
p) MOD 256
2450 IF by=0 THEN 2480
2460 FOR b=0 TO by-1:BPUT#t,page?b:NEXT
2470 VDU31,s DIV 10,s MOD 10+7,251
2480 s=s+1:NEXT bk
2490 CLOSE#t:cmd?p=32:*DISC
2500 VDU23;11,255;0;0;0:ENDPROC
2510 :
2520 DEFPROClock(n,s$)
```

→38

MAPPING THE BRITISH ISLES

Grahame Blackwell shows how to map out the British Isles with his useful routine. And you can change the screen position and scale as well.



This useful program will draw the British Isles at any size, in any position and at any aspect ratio in any graphics mode on the BBC micro. It uses completely conventional technology (!), and should therefore work on any BBC micro configuration. It is also easily imported into other programs. In part two of "Painting by Numbers" in this issue, a routine is given to fill the land in green and the seas blue; and in a forthcoming article on printer graphics, we will be using this drawing in our Page-maker feature.

First of all type in the listing carefully - if you get the data statements wrong the map will inevitably suffer. As a precaution, save the program before you run it. When it is run, the program asks for a number of parameters. The X and Y scales can be any value that you like: a value of 4 for each will give a full sized map in any mode. By giving the X scale a different value to the Y scale, you can change the aspect ratio of the map. With X=4, and Y=2, you get a map reminiscent of views from space satellites.

The next two parameters prompted for are the co-ordinates of the origin. Pressing Return without giving a value will enter zero, and this positions the map centrally for a full-size drawing. Small maps on this setting will appear in the bottom left hand corner of the screen. Lastly the mode is requested. Any graphics mode (0, 1, 2, 4 or 5) may be entered here. The mode will then be set, and the map drawn.

Depending on how you intend to use the map, you will probably wish to alter line 200. As you can see, the whole drawing operation is run from line 190, calling two procedures on the way. Line 200 carries the END statement. If you wish to simply display the outline map on the screen, you can replace it with:

```
A=GET
```

This will hold the screen until a key is pressed. Alternatively you may wish to call a screen dump, either to printer or to disc or cassette at this point.

In case you are miffed by the structure of the data statements in the program, they contain pairs of X-Y co-ordinates in hex format, and are read one line at a time into A$, before being decoded by PROCpoint, and drawn by PROCisland.

```
   10 REM PROGRAM BRITISH ISLES
   20 REM Version B0.6
   30 REM Author  G.Blackwell
   40 REM BEEBUG  Aug/Sept 1986
   50 REM Program subject to copyright
   60 :
  100 MODE7
  110 PRINT'CHR$141;CHR$134;SPC6;"BRITIS
H ISLES MAP"
  120 PRINTCHR$141;CHR$134;SPC6;"BRITISH
 ISLES MAP"'''''
  130 INPUT"X-Scale (4=full size):    "W
  140 INPUT"Y-Scale (4=full size):    "H
  150 INPUT'"Horiz Position (0-1279):  "
PX
  160 INPUT"Vert  Position (0-1023):  "P
Y
  170 INPUT'''"Mode: "M:MODE M
  180 VDU29,PX;PY;
  190 REPEAT PROCisland: UNTIL Y=0
  200 END
  210 :
  220 DEFPROCisland
  230 READ A$:PROCpoint:MOVE X*W,Y*H
```

```
240 REPEAT IF A$="" READ A$
250 PROCpoint:IF X*Y>0 DRAW X*W,Y*H
260 UNTIL X*Y=0
270 ENDPROC
280 :
290 DEFPROCpoint B$="&"+LEFT$(A$,4)
300 A$=RIGHT$(A$,LENA$-4):Y=EVALB$
310 X=Y DIV256:Y=Y MOD256
320 ENDPROC
330 :
340 DATA 9CF69AF29CEE8CDE91DE8EDA96DC
350 DATA 9ADCA4DAAADBAEDAB0D6AED2A6BC
360 DATA A2BA9EBAA4B498AD92AE96AAA0AD
370 DATA AAAAB3A1BC89BA86C086CE7ACC76
380 DATA D26CC86CD268D660D65DD058D656
390 DATA DC5AE65AEC56EE50EC42E83EE83A
400 DATA E23AE238E335E132DA30E230E42E
410 DATA EC30EC28E626E421D81BD41CC81C
420 DATA C41AC41EC01EC01CBC1EBC1CB419
430 DATA B01AB417B214A616A21A9A169816
440 DATA 981294ØB900B8A0E800E800B7A08
450 DATA 7C05760573086E056E0A780E7E16
460 DATA 8218841D84218A218A2696289C26
470 DATA A4309E309A2B922D92328E328A30
480 DATA 86367E327A3276367A38763A8C46
490 DATA 8E508C56845183527F52885A885C
500 DATA 906094609861A05E9C659C6AA46E
510 DATA 9D6E9E73A072A27A9E7A9C769A76
520 DATA 9A7B94829A8F968F948C8E8A888C
530 DATA 8886808C80857E857A907D8E849E
540 DATA 80A480AC7CA478AA78A474987098
550 DATA 74AA76BA70B46AB46CB86ABC6EBE
560 DATA 6CC172C274CB70C964D466DA6AD6
570 DATA 6ADC72D072D674E07CE079E57CE6
580 DATA 7AEA7EEA7CEE7EF484F287F489F2
590 DATA 8CF496F497F69CF60001
600 DATA 70F06CEA6AE262DC60E062E360EA
610 DATA 64EA66EE68EE6CF270F00001
620 DATA 5FDB5FD65DD65CCA56C458D456DC
630 DATA 5FDB0001
640 DATA 70B06FB06BAA66A66AA06EA26EA5
650 DATA 71B00001
660 DATA 7AA07DA27E9C7B9C7AA00001
670 DATA 827482798880897A82740001
680 DATA 865E8C62866584628650D001
690 DATA BE1AC218BC16BA18BE1A0001
700 DATA 6A97728B6E867586768074807182
710 DATA 727E747C707A6D7666736A696962
720 DATA 6B5A684C64466642524146384 43A
730 DATA 4035333130332832 2E3726362A3C
740 DATA 263B213C22402A44204524492A47
750 DATA 2A4C325028503256325A345E3A5E
760 DATA 3A6537622E642C672668286C2C6C
770 DATA 2D713171327 42C7428772D77297B
780 DATA 2A7F2E7E2E81387F3A7C3C7E437C
790 DATA 428048824A8746874089448B448E
800 DATA 488E4C8C539856945699599C5E9A
810 DATA 5C995A945C915F9360986A970100
```



FILLING THE BRITISH ISLES

The British Isles map listed in this issue, can easily be filled using the filler routine given last month. The simplest way to accomplish this would be to fill the sea area using the fill routine, and then use VDU19 to make the sea blue, and the land green. But to illustrate the way in which multiple fills can be used, I will do it the hard way, filling first the sea area, and then each one of the enclosed land masses individually.

The program to achieve this for a map drawn centrally to full size (i.e. scale 4,4; position 0,0) is listed below. It should be added to (or merged with) the map-drawing program. Of course, it makes use of the machine code fill routine, and assumes that this has been saved as FILL (line 910), and that its execution address is &900, as discussed last month.

Line 200 replaces the END statement in the drawing program, and steers the program to line 900, where the fill routine begins. Lines 930 and 940 set suitable colour allocations so that the colouring to the land and sea are correct for all graphics modes. Lines 950 - 970 fill the sea area, while the FOR loop on line 980 calls the fill routine a further 9 times to fill each land mass. This includes Anglesey, which has somehow been cut off from the mainland to a similar extent to the Isle of Wight!

```
 200 GOTO 900
 900 REM BRITISH ISLES FILLER Ver B0.5
 910 *LOAD FILL
 920 paint=&900
 930 VDU 19,7,3;0;19,3,3;0;19,1,4;0;
 940 VDU 19,2,2;0;:
 950 A%=&FFFFFF01:B%=&FFFFFFFF
 960 C%=B%: D%=B%
 970 X%=0:Y%=0:CALL paint:A%=&FFFFFF02
 980 FOR I=1 TO 9
 990 READ X%,Y%:CALL paint
1000 NEXT
1010 DATA 520,920,400,920,368,864
1020 DATA 424,664,500,632,532,488
1030 DATA 540,392,756,96,320,400
```

This concludes discussion of the applications of FILLER. For those of a technical bent, the article ends with a discussion of how FILLER works.

PAINTING BY NUMBERS (Part 2)

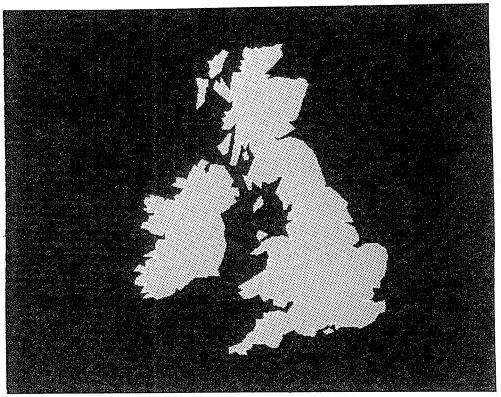
Grahame Blackwell fills in the details of his flexible paint routine, useful for colouring in maps like that of the British Isles opposite.

HOW FILLER WORKS

FILLER uses the principle of recursion to handle twists or forks in the fill area - one direction is followed, whilst others are noted for completion later; you may have observed this effect when running the Spiral demonstration program listed last month. Pending operations are held on the CPU stack, which is more than adequate for any normal situation. In the case of infinite recursion, which could happen by repeated overfilling (as may occur if boundary colours are incorrectly set), the routine incorporates a cut-out to prevent the stack overfilling and crashing the program. In such a case, the routine does not automatically terminate, it simply limits the number of pending operations at any time (always leaving several bytes of CPU stack for OS event-handling interrupt routines).

For speed, the routine operates directly on screen memory, and will therefore not operate across the tube.

The main sequence of operations for an arbitary area fill is:



(a) Fill a line of pixels left and right from given co-ordinates to boundary pixels or the edge of the screen.
(b) Search for possible continuations of the fill, and note each continuation on the stack for attention later.
(c) Check stack for entries. If no entries, terminate the process; otherwise, remove the last stack entry for processing, and perform (a), (b), (c).

Each sequence of steps (a), (b) may generate just one stack entry, for simple continuation in the same direction, or two or three if the fill branches and/or loops back. This particular routine enters such sub-fills on the stack last, and thus deals with them before un-stacking and continuing the main fill direction; each sub-fill becomes a fill sequence in its own right, possibly leading to more branches and loops - this is the recursion mentioned previously, essential to any successful arbitrary area fill.

The subroutines in FILLER performing steps (a) and (b) are rout3 and rout6, controlled by rout7, which corresponds to operation (c). They in turn use:- rout1, which locates the pixel corresponding to particular screen co-ordinates, reads the colour of that pixel, and finds what the new colour of that pixel should be; rout2, which changes the colour of a pixel; rout4 and rout5, which move one pixel to the left and right respectively. Lines 140-230 check the mode (exiting if non-graphics), and transfer start co-ordinates and mode-dependent data to zero page. Lines 240-310 set up the colour mask table from A%-D%, each entry held as for one pixel in the current mode. Lines 320-360 set up a fast fill mask for a complete zero screen byte, fill the first line, and enter rout7 ready to fill in both directions.

MEMORY USEAGE

As discussed in last month's article, FILLER is assembled to &900, and takes just under 3 pages of memory. It is a simple matter to assemble the code anywhere else in memory, as is convenient. In addition to the three pages of memory used to store the routine, FILLER uses the zero page locations from &70 to &8F as workspace. Other software sharing this workspace will need to reinitialise after FILLER has been called.

INTER-WORD

FROM

Computer Concepts

Computer Concepts made their name with Wordwise, one of the most popular and successful word processors for the Beeb. Now Inter-word has arrived from the same stable. Geoff Bains has been examining the import of this major new release.

Product : Inter-word
Supplier : Computer Concepts
: Gaddesden Place,
: Hemel Hempstead HP2 6EX.
: Tel. 0442-63933
Price : £56.35

At long last Inter-word has arrived. As supplier of one of the most popular word processor for the BBC micro - Wordwise - Computer Concepts has a lot to live up to in its successor. Fortunately, the new Inter-word seems to be just about everything anyone could ask for in a word processor for the Beeb.

Inter-word is supplied on a 32K ROM mounted on a tiny carrier board. To the Beeb it appears just like any single 16K ROM but in fact it has twice as much code inside. This ROM and board do seem to get a little warm but the system is a major improvement on Acorn's already excellent sideways ROM system.

To start with Inter-word looks remarkably like Wordwise. After calling up the program with *IW. a mode 7 menu appears. This offers the options of saving or loading the entire document, saving a marked section only, inserting text at the cursor position, setting up the printer, printing the text, previewing the text, spooling the text, and of course entering the editor itself.

This menu is very like that in the old Wordwise. However, once a selection is made things are immediately different. All the filing operations produce an immediate catalogue of the disc in the current drive (cassette systems excepted, of course). The filename to use may then be selected from the catalogue as an item in a menu, using the cursor keys. The menu item selected by default is the last used name. Alternatively the filename for use can be typed in from the keyboard. Indeed, a combination of methods may be used and an existing name edited to create another.

When saving or spooling a file a 'Replace old file Y/N?' prompt is given if the filename is already used. When loading, a check is made that any existing text is to be overwritten. Such an interface with the filing system (any Acorn compatible filing system, too) is a major work in itself. However, the real joy of Inter-word is in the editing.

Unlike any other major word processor for the Beeb, Inter-word is a genuine 'what you see is what you get' word processor. Text appears on the screen EXACTLY as it does on paper. Entering text is easy enough. You just type it in. Naturally Inter-word provides a word wrap facility so that you can just keep on typing and not worry about new lines.

When a new line is required 'out of place' the Return key is used. This forces a new line and is shown on the screen with a bent left-pointing arrow. As your typing fills a page, the page boundary is displayed and the cursor automatically leaps over this to the first line of the new page. Deleting and adding text can be a bit disconcerting at a page boundary. The cursor is prone to leaping back and forth. However, this just shows Inter-word's impressive speed and accuracy.

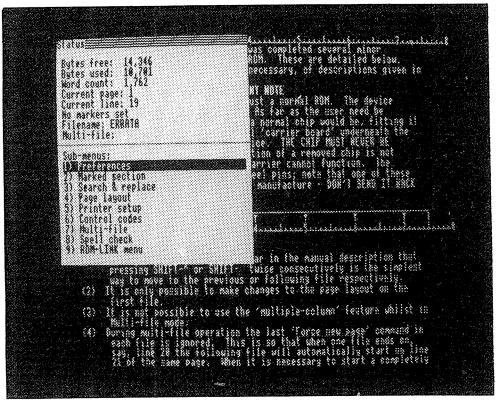
As you type in text you can go back over it and alter it. Text can be written in 'insert' or 'overwrite' modes and the delete key has its normal function. A letter can be deleted at the cursor position by pressing either Ctrl-A or the Copy key. A word can be deleted with Ctrl-D and a whole line with Ctrl-L.

Moving around the text to make the alterations is similarly catered for. The cursor keys alone, of course, move the cursor a single letter at a time along a line and a line at a time vertically. Pressing Ctrl along with the cursor keys moves by the word horizontally and by the screenful vertically. Shift with the cursor keys moves to the extremes of the lines horizontally and to the top and bottom of the document vertically. This is all compatible with Wordwise. In addition,

Ctrl-P and Ctrl-N will move to the same place in the previous or next page.

Naturally, text is not restricted to the arrangement you give it as you type. Text can be justified, or aligned to the left or right margins, or centred on the line. Again, altering an already centred line can be a little weird because Inter-word re-centres it as you type.

The arrangement of text on a line is altered with the function keys. Unlike Wordwise, four of these have each been given two functions distinguished by pressing the function key along with the Shift key. However, the function key strip is still reasonably uncluttered unlike that of View and many of the newer word processors that have sprung up recently (see BEEBUG Vol.5 No.1).



To centre or underline a piece of text, it is first marked in much the same way a piece of text is marked in Wordwise or View. This done, pressing one of six function keys (four with or without the Shift key) will align the section left or right, centre it, justify it, delete it, copy it to the cursor position, underline it, set it in bold text, or perform a definable function that is shown on the screen as a dotted underline and appears on the printout as whatever effect you care to define (default is Epson italics).

The format of the pages is controlled with rulers in a very similar way to View. Wordwise's embedded codes for this have been totally thrown out. A ruler is inserted anywhere in the text by pressing another function key and it's then edited directly on the screen.

The left and right margins and the tab positions are dragged to the desired position with the Shift and cursor keys as is the 'Return margin'. The first character typed after a Return is pressed (at the start of a new paragraph) can be placed at a different position to the first characters of the other lines, giving automatic paragraph indenting.

The other facilities of Inter-word are controlled using 'pull-down' menus. In my view, this is a bit of a shame. When Inter-word was started (many months ago) pull-down menus were all the rage and popped up in every software package imaginable. It is true that pull down menus provide a means to make selections that is very easy to get to know. The beginner loves them. However, after you become more used to using a piece of software the craving for a quicker, if less friendly method, becomes greater. However, it is with pull-down menus that Inter-word is provided, and within these limitations they perform very effectively.

There are nine selection menus. These are all accessed through a tenth that is called from the keyboard by pressing f0. The menu selection menu also shows the status of the word processor - the number of bytes free and used, word count, current page, line, and other information. The lower half of this pull-down box forms the menu for the other menus. The method of selecting the desired option is made as simple and flexible as possible. The vertical cursor keys are used to highlight one of the menu items and then Return selects it. Alternatively, either the function keys or the number keys can be used to select the corresponding item.

The first menu is the preferences menu and this alters aspects of the use of the word processor. The number of lines displayed on the screen can be altered as can the colours used for text and background. The key repeat rate and cursor blink rate can be altered and even the interlace switched on or off. The most important selection is the number of columns displayed. As you'd expect 40 or 80 columns can be displayed but also 53 or 106 columns. This is achieved by a complete redefinition of the character set giving thin but readable characters. In

fact the ruler margins can be extended beyond the displayable limits and then a kind of sideways scrolling operates.

The second pull-down menu covers those operations that can be carried out on a marked section - duplicates of the function keys. The third menu covers the search and replace operation. This can be configured (from the menu, naturally) to ignore case differences and to confirm changes at each find. Wildcards are allowed in the search string. A search operation can also be started without resorting to the pull-down menus by pressing Ctrl-G.

The Page layout menu lets you set up page length, top and bottom spaces, and position of headers and footers. The headers and footers themselves are defined in the main text area. A marked section of text can be defined as a header/footer text and then called up when required with an embedded code. Unlike Wordwise these codes take up no room on the screen. To position an embedded code another menu is selected with f1. Codes to force a new page, or issue a * command are all possible but are shown in the text as a reverse video character. Once a section of text has been defined as a header/footer it can be deleted from the text area.

The printer set-up menu deals with the number of copies and which pages of the document are to be printed, line spacing, pad characters, the codes for the underline, bold and user definable effect, and the like. Each menu seems to have its high point and the special feature in this one is the column printing. Inter-word can print text in up to five columns, each column on a sheet of paper taken from the next page in the text. Using this facility the two column format used in BEEBUG could be simply reproduced in one printing operation. It is for column printing that the preview option in the main menu is provided as this is the only effect that cannot be seen as the text is edited.

The remaining three menus are very special. The multi-file menu deals with a very clever feature of Inter-word. A piece of writing that is too long to fit in memory can be split into several files, named as you require. These can be printed continuously with only one order to print and when editing you can jump from one to another by pressing Shift along with a vertical cursor key, without giving the next file's name. Inter-word looks after all of that.

Computer Concepts are planning a spelling checker to accompany Inter-word. The spelling check menu (already built into the word processor) can then be used to start a check on your document or initiate a continuous check as you type.

The last menu is the ROM-link menu. This deals with Inter-word's interaction with other 'Inter' ROMs in Computer Concepts' series. The ROM link idea allows up to 16 separate 'packages' of data for each ROM-link program to share the Beeb's memory at any one time. The ROM link menu displays the packages in memory at the time and allows you to enter any one of these directly. The system also enables you to 'import' and 'export' data between the different ROM-link programs.

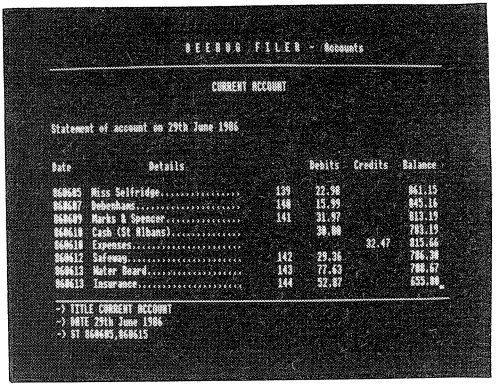
Getting to know any comprehensive piece of software such as a word processor is bound to be a long process. However, the effort required to learn Inter-word is definitely worthwhile. The manual accompanying Inter-word is of a good standard; though it would have been nice to have seen an 'Introduction to Inter-word' booklet along the same lines as provided with Wordwise Plus.

Inter-word is the most powerful and comprehensive word processor available for the Beeb today. It is more powerful and flexible than View but still retains much of the friendliness and ease of use of Wordwise. Inter-word looks set to become the standard word processor for the BBC micro: a role to which it is eminently suited.

BEEBUG is offering Interword to members at the special price of £48.00 inc. VAT and p&p. Existing Wordwise Plus users are entitled to a further discount of £3 (see supplement for details).

FILER ACCOUNTS OPTION

Mike Williams further extends the BEEBUG Filer database system by showing how this can be used as the basis of a home accounts package.



The latest addition to the BEEBUG Filer database system, first described in BEEBUG Vol.4 Nos.6, 7 & 8, is designed to provide an effective, easy-to-use home accounts package. That is, it will allow you to use the computer to keep track of a bank account, or similar, recording details as you enter them of all your credits and debits, and providing a statement or balance of account when requested.

The system described here uses the original BEEBUG Filer to create and maintain one or more bank files. Once a bank file has been set up you continue to use Filer to enter details of your cheques and other payments, and details of any credits. All the facilities of Filer can be used, as required, to search, sort and print out records from this file.

The program listed this month is a quite separate program, though following the same general Filer format, which will take the information entered in your bank file and compile this into a bank statement, or display an up-to-date balance of your account. You could also use the Filer Graph Option in BEEBUG Vol.5 No.2 to display the state of your account graphically, maybe too graphically for some!

SETTING UP A BANK FILE

Use BEEBUG Filer to create your bank file. You will need one record for each possible transaction, but the Filer Accounts Option allows you to close one bank file and carry forward the closing balance to start a new bank file. Thus you may wish to set up a file to cover a month or maybe a quarter at a time. When specifying the structure of the file the following fields MUST be included with the field widths shown. Other fields may be included if you wish and all fields may be in any order.

Fieldname	Fieldwidth
DATE	6
DESCRIPTION	30
CHEQUE	6
AMOUNT	8
BALANCE	8

The field names MUST be exactly as given above. You will also need to create a second file with the same name as the first but with directory P (e.g.P.MYBANK). This file will eventually be used to hold details of your direct debits and other regular payments, although this facility will only be fully implemented later. This file is an essential part of the Filer Accounts system and should be created with the following fields only, and in the exact order given:

Fieldname	Fieldwidth
DES	30
AMOUNT	8
FREQ	2
DATE1	6
DATE2	6

Once this 'Payments' file has been created it can be ignored, though used always by the Filer Accounts program. Details of how to use this file for direct debits and the like, and the code to implement this, will be given in the next issue.

Finally, to start your bank file, you will need to add one record to it to specify your opening balance. Include the date as well, but leave all the other fields empty except for the balance itself.

Every time you write a cheque or make any other kind of payment or deposit affecting your account you should add another record to your bank file using Filer. Dates should always be in the form 'yymmdd', that is two digits for the year, two for the month, and then two final digits for the day. Entries should be maintained in date order as far as possible.

The amount to be debited should be entered as a money sum, i.e. with two digits always following the decimal point. Since this is effectively a file of debits, any credits should be entered as negative numbers in the same 'money' format. The cheque and description fields may be filled in as you wish - they are not processed but are just listed on your statement. The balance field should always be left blank as this is computed and filled in automatically later.

USING THE FILER ACCOUNTS OPTION

Type in and save the program listed at the end of this article. For Basic I, replace all occurrences of 'OPENUP' by 'OPENIN' (see lines 3060, 3420, 5300, 5360). The program will run with PAGE at its default of &1900, but to maintain compatibility with the other Filer programs and allow for further additions you are again recommended to run the Accounts option with PAGE set to &1400 (unnecessary on the Master, and B+ with shadow memory - use MODE 131 at line 100).

The commands recognised by Filer Accounts are:

```
OPEN          <filename>
CLOSE
TITLE         <title>
DATE          <date>
STATEMENT     <date1>,<date2>{/P}
BALANCE
CF            <filename>
COMMANDS
END
```

These commands are used as follows:

OPEN Open your main bank file for use - automatically opens the payments file as well.

CLOSE Close the bank file, and of course the payments file.

TITLE Specify any title to appear at the top of your bank statement.

DATE Specify, in any format, the date of your statement.

STATEMENT Request a statement to be produced covering the period from the first date up to the second date. If the first date is omitted the statement will start with the first record in your bank file. If the second date is omitted the statement will continue up to the date of the last record in the file. Appending '/P' will cause the statement to be sent to your printer. All three parameters are optional.

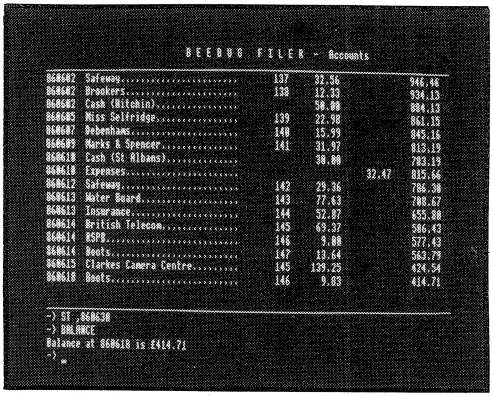
BALANCE Request an up-to-date balance of account, displayed on the screen only.

CF Close the current bank file, carrying the closing balance forward to form the first record of the new file specified. The payments file will also be updated and renamed to reflect this.

COMMANDS List all valid commands.

END Exit from the program, closing all files.

An example of a statement produced using the Filer Accounts Option is included with this article. Once set up, the Filer Accounts Option is simplicity



itself to use, and the implementation of direct debits, to be described in the next issue, will complete a most useful application of the Filer Database system.

PROGRAM NOTES

Space does not permit a detailed description of the program, but many of the procedures used are the same or similar to ones used previously with Filer, and the general structure of the program is also similar. Do keep to the line numbering in the listing when typing in the program. Note also the value of the variable FDR in line 1060. This sets the size of the File Description Record allowing this to be readily changed by anyone who has modified or extended the original Filer Database program.

The value 123 in line 3480 is also important. You can always force the program to re-initialise the payments file and re-calculate all balances by setting this temporarily to any other value.

The magazine cassette disc also contains a useful menu program (FMENU) that you can use to switch easily between the three Filer programs (Database Manager, Graph Option and this month's Accounts Option). These should be set up on a single disc with a BOOT file which will set PAGE to &1400, initialise any printer and chain in the menu program. In all three Filer programs, change line 160 to read: 160 PROCclose:CH."FMENU".

```
  10 REM BEEBUG FILERA version B1.0
  20 REM Author Mike Williams
  30 REM BEEBUG AUG/SEPT 1986
 100 MODE3:ON ERROR PROCerror:END
 120 PROCsetup:PROCtitle:PROCwindow2
 140 REPEAT:PROCcommand:UNTIL exit%
 160 PROCclose:VDU26:*FX4,0
 180 END
 200 :
1000 DEF PROCsetup
1020 LOCAL I:exit%=FALSE:open%=FALSE
1040 maxf=12:X=0:Y=0:date$="":title$=""
:*FX4,2
1060 debit%=FALSE:FDR=512
1080 DIM com$(10),record$(maxf,1),field
$(maxf),width%(maxf),os 40
1100 READ N
1120 DATA 9
1140 FOR I=1 TO N:READ com$(I):NEXT I
1160 DATA OPEN,CLOSE,TITLE,DATE
1180 DATA STATEMENT,BALANCE,CF,COMMANDS
,END
1200 ENDPROC
1220 :
1300 DEF PROCtitle
1320 PRINTTAB(0,1)STRING$(80," ")
1340 PRINTTAB(28,1)"B E E B U G  F I L
E R  -  Accounts"
1360 PRINT STRING$(80,"_")
1380 PRINTTAB(0,20)STRING$(80,"_")
1400 ENDPROC
1420 :
1500 DEF PROCcommand
1520 LOCAL command$,pm$
1540 REPEAT
1560 INPUT LINE"-> " command$
1580 C=FNvalidate(command$)
1600 IF C=0 THEN PRINT"Unrecognised com
mand"
1620 UNTIL C
1630 IF C=1 THEN PROCopen(pm$)
1640 IF C=2 THEN PROCclose
1650 IF C=3 THEN title$=pm$
1660 IF C=4 THEN date$=pm$
1670 IF C=5 THEN PROCstatement(pm$)
1680 IF C=6 THEN PROCbalance
1690 IF C=7 THEN PROCcarry(pm$)
1700 IF C=8 THEN PROClistc
1860 IF C=9 THEN exit%=TRUE
1870 IF C=99 THEN PROCstar(pm$)
1880 ENDPROC
1900 :
2200 DEF FNvalidate(C$)
2220 LOCAL found%,I,P:found%=0
2240 FORI=1 TO N
2260 IF (ASC(MID$(C$,1))AND223)=ASC(MID
$(com$(I),1)) AND (ASC(MID$(C$,2))AND223
)=ASC(MID$(com$(I),2)) THEN found%=I
2280 NEXT I
2300 P=0:IF C$>"" THEN P=INSTR(C$," ")
2320 IF P=0 THEN pm$="" ELSE pm$=MID$(C
$,P+1)
2340 IF ASC(C$)=42 THEN pm$=MID$(C$,2):
found%=99
2360 =found%
2380 :
3000 DEF PROCopen(p$):LOCAL I
3020 IF p$="" THEN PRINT"No file given"
:ENDPROC
3040 IF open% THEN PROCclose
3060 F$=p$:F=OPENUP(F$)
3080 IF F=0 THEN PRINT"No such file":EN
DPROC ELSE open%=TRUE
3100 PTR#F=0:INPUT#F,rec,recn,recs,f
3120 FOR I=1 TO f:INPUT#F,field$,width%
(I):field$(I)=FNstrip(field$,"."):NEXT I
3140 date=FNfield("DATE"):chq=FNfield("
CHEQUE"):desc=FNfield("DESCRIPTION")
3160 class=FNfield("CLASS"):bal=FNfield
("BALANCE"):amnt=FNfield("AMOUNT")
3180 PROCpayments:ENDPROC
3200 :
3400 DEF PROCpayments:LOCAL I,p1$,p2$
3420 F1$="P."+F$:F1=OPENUP(F1$)
3440 IF F1=0 PRINT"No payments file":PR
OCclose:ENDPROC
```

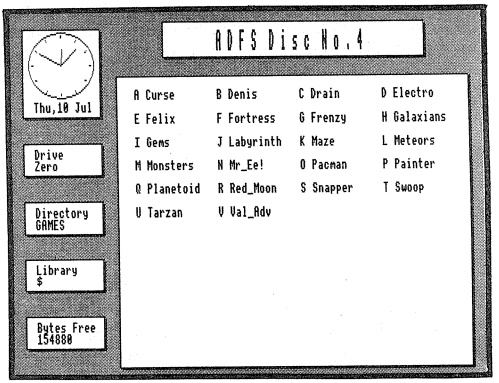
```
 3460 debit%=-1:INPUT#F1,recp,I,I,pf
 3480 PTR#F1=FDR-1:I=BGET#F1:IF I<>123 P
ROCinitialise
 3500 IF NOT debit% PROCclose:ENDPROC
 3520 PTR#F1=FDR-25:INPUT#F1,recb,p1$,p2
$:db=VAL(p1$):balance=100*VAL(p2$)
 3540 ENDPROC
 3560 :
 3700 DEF PROCinitialise:LOCAL p1$,p2$
 3720 PRINT"Initialising payments file"
 3740 IF rec=1 PRINT"No entries in bank
file":debit%=0:ENDPROC
 3760 PROCread(1,F,0):p1$=record$(date,0
):p2$=record$(bal,0)
 3780 PTR#F1=FDR-25:PRINT#F1,1,p1$,p2$:d
ebit%=-1:PTR#F1=FDR-1:BPUT#F1,123
 3800 ENDPROC
 3820 :
 4000 DEF PROCbalance
 4020 IF NOT open% PRINT"No file open":E
NDPROC
 4040 IF rec>recb+1 PROCsheet(recb,db,99
9999,FALSE,FALSE)
 4060 PRINT"Balance at ";STR$(db);" is `
";STR$(balance DIV 100);".";RIGHT$(STR$(
balance),2)
 4080 ENDPROC
 4100 :
 4200 DEF PROCstatement(p$)
 4220 LOCAL d1,d2,p,:print%=FALSE
 4240 IF NOT open% PRINT"No file open":E
NDPROC
 4260 p=INSTR(p$,"/P"):IF p print%=-1:p$
=LEFT$(p$,p-1)
 4280 p$=p$+",":p=INSTR(p$,","):d1=VAL(L
EFT$(p$,p-1)):p$=MID$(p$,p+1)
 4300 p=INSTR(p$,","):d2=VAL(LEFT$(p$,p-
1))
 4320 IF d2=0 THEN d2=999999
 4340 IF d1<=db THEN r=FNchop(d1,recb) E
LSE r=recb
 4360 PROCsheet(r,d1,d2,print%,TRUE)
 4380 ENDPROC
 4400 :
 4500 DEF PROCsheet(r,d1,d2,p1%,p2%)
 4520 PROCwindow1:CLS:IF p1% VDU2 ELSE V
DU14
 4540 REPEAT:line=10:IF p2% PROCsheet1
 4560 PROCread(r,F,0)
 4580 REPEAT
 4600 IF r>recb balance=balance-100*FNva
l(amnt):record$(bal,0)=STR$(balance DIV
100)+"."+RIGHT$(STR$(balance),2):PROCwri
te(r,F,0) ELSE balance=100*FNval(bal)
 4620 IF p2% AND  FNval(date)>=d1 PROCpr
intline(record$(date,0),record$(desc,0),
FNstrip(record$(chq,0),"."),FNstrip(reco
rd$(amnt,0),"."),FNstrip(record$(bal,0),
"")):line=line+1
 4640 r=r+1:IF r<rec PROCread(r,F,0)
 4660 UNTIL r>=rec OR FNval(date)>d2 OR
line>60
 4680 IF p1% VDU1,12
 4700 UNTIL r>=rec OR FNval(date)>d2
 4720 VDU3,15
 4740 PROCread(r-1,F,0):PTR#F1=FDR-25:PR
INT#F1,r-1,record$(date,0),record$(bal,0
)
 4760 recb=r-1:db=FNval(date):balance=10
0*FNval(bal):PROCwindow2:ENDPROC
 4780 :
 4800 DEF PROCsheet1
 4820 PRINT'SPC(40-LEN(title$)DIV2);titl
e$
 4840 PRINT''"Statement of account on ";
date$
 4860 PRINT''"Date";TAB(20);"Details";TA
B(52)"Debits";TAB(61)"Credits";TAB(71);"
Balance"'
 4880 ENDPROC
 4900 :
 5000 DEF PROCprintline(date$,desc$,chq$
,amnt$,bal$):LOCAL P
 5020 IF VAL(amnt$)>0 THEN P=49 ELSE P=5
9:amnt$=MID$(amnt$,2)
 5040 PRINT date$;TAB(8);desc$;TAB(40);S
PC(8-LEN(chq$));chq$;TAB(P);SPC(9-LEN(am
nt$));amnt$;TAB(69);SPC(9-LEN(bal$));bal
$
 5060 ENDPROC
 5080 :
 5200 DEF PROCcarry(p$):LOCAL I
 5220 IF p$="" PRINT"No file given":ENDP
ROC
 5240 IF NOT open% PRINT"No file open":E
NDPROC
 5260 PROCbalance:PROCread(recb,F,0):CLO
SE#F1
 5280 PROCoscli("SAVE "+p$+" 0+"+STR$~(F
DR+recn*recs))
 5300 I=2:F2=OPENUP(p$):PTR#F2=0:PRINT#F
2,I,recn,recs,f
 5320 FOR I=1 TO f:PRINT#F2,field$(I)+ST
RING$(12-LEN(field$(I)),"."),width%(I):N
EXT
 5340 record$(chq,0)=STRING$(width%(chq)
,"."):record$(desc,0)="Balance carried f
orward"+STRING$(width%(desc)-23,"."):rec
ord$(class,0)=STRING$(width%(class),".")
:record$(amnt,0)=STRING$(width%(amnt),".
"):PROCwrite(1,F2,0):CLOSE#F2
 5360 F1=OPENUP(F1$):PTR#F1=FDR-25:PRINT
#F1,1,record$(date,0),record$(bal,0)
 5380 PROCclose:PROCoscli("RENAME "+F1$+
" P."+p$)
 5400 PRINT"Balance carried forward to "
;p$
 5420 ENDPROC
 5440 :
 6000 DEF PROCclose
```

→39

MASTER SERIES

An ADFS Menu

Just for Master users, Peter Rochford has produced a stunning ADFS menu program providing easy access to all your programs and files.



The ADFS provides a good many useful facilities for Master users, and its tree-like file structure is a great idea. But using it is very hard work. This utility is intended to make life a little easier (aren't they all?). It is essentially a menu-driven front end, allowing easy movement through a disc's directories, and subsequent selection of files to be CHAINed or *RUN etc. It has been designed specifically for the Master and Master Turbo, and makes use of the built in clock/calendar to display a small analogue clock which happily ticks away the seconds in the corner of the main screen, even though the whole thing is handled in Basic.

The menu, which uses mode 128 to give a Macintosh look-alike front end, reads all the data from your disc automatically. All you have to do is to save the menu to the root directory of your disc (or discs), and put on an appropriate !BOOT file to ensure auto-boot. Then when the disc is booted you will be presented with the title of the root directory in double height, and small windows on the left will show the current drive, current directory, current library, and the number of bytes free on the disc.

The large window provides a menu of designated items in the current directory; and has sufficient space for the full allocation of 47 objects. Any one of these may then be selected at a single key-press. If the item is a program file, it will be CHAINed or *RUN as appropriate, but if it is a directory, the display will be updated to show the contents of the new directory; again with directory title in double height, and with menu-selectable contents in the main window below. You can repeat this process to any depth of nesting that you wish - the menu will automatically follow.

Pressing the UP cursor key at any time will take you to the root directory, while the LEFT cursor key will take you back to the previously displayed directory. To change drives, enter "1" or "Ø" as appropriate. You may also execute any star command from any menu screen, though of course if you use any which over-write the program you will not be able to get back to the menu. To help you out here, and to allow the menu to be recalled even after CHAINing a program from it which over-writes it, function key 9 should re-boot the system.

In fact the menu will recognise three types of files:

1. Basic programs to be CHAINed
2. Machine code programs to be *RUN
3. ROM images to be loaded directly into sideways RAM.

In the case of the ROM image loader, you will need to press Ctrl-Break to initialise any ROMs loaded. The menu prompts you to do this, but you will not be able to get directly back to the menu after this. The easiest thing to do is to re-boot the disc with Shift-Break.

The menu reads directory names, titles and filenames straight from the disc; and in the case of the latter, it distinguishes between the different types of file by looking at their load addresses. This is necessary because, although it is appropriate to use the directory on the DFS to indicate the type of file for an auto-menu, this is not feasible on the ADFS, since it would

drastically limit the user's choice of directory names. The only important occasion where the program can be confused over what to do with a file is when machine code has been tacked on to the end of a Basic file, as occurs in some games. A way of dealing with this is outlined at the end of the article. One further point is that ROM images must have a load and execution address of &8000 if the menu is to recognise them.

INSTALLING THE SYSTEM

First of all, type in the program carefully, and save it away to disc under the ADFS in the $ directory with the title "menu". You will find it helpful in general to distinguish filenames from directory names by using upper case for directories, and lower case, or mixed upper and lower for filenames. As you will see the program has been presented in a somewhat compact fashion. This has been done in order to keep it fast and economical on memory. When you have typed it all in, you should create a boot file with the contents shown in the accompanying listing. This is easily done with Edit or by using *BUILD, and the file should be saved as $.!BOOT, again on the ADFS. Now type:

*OPT4,3

to set the boot option on the disc; and Shift-Break will then boot up the menu.

The way that the menu recognises whether to include a file or directory in its display is by checking for its ACCESS status. If it is set as R (read only), or RL (read-only and locked), then it will be included in the display. To set the required option on files that have been saved to the disc you will need to perform:

*ACCESS <filespec> R (or LR)

You can of course use:

*A. * LR

to speed things up. In the case of a game which has three component programs, such as Acornsoft's Snapper, you could ensure that only the first of the three appears on the menu by giving the other two RW status.

TECHNICAL NOTES

The menu program is not run from the normal default PAGE setting of &E00; but from &6000. This is set up in the boot file. The reason for this is to allow the ROM image loader to work in the normal user memory area, while leaving the menu in tact. Also, programs that are CHAINed in by the menu may also be able to function without disturbing the menu; though to recall it you would need to reset PAGE to &6000 before typing RUN. It is probably easier in this case to re-boot the menu using function key 9.

HOW TO ALTER HYBRID FILES

Basic files with machine code tacked on to the end must be adjusted as follows: Enter *INFO <filename>, and note down the last 4 digits of the penultimate number in the resulting display. This is the length of your file. Now make use of this as follows:

*LOAD <filename> E00 <return>
*SAVE <filename> E00+<length> 802B <return>
*ACCESS <filename> R <return>

And the job is done.

!BOOT File

```
*BASIC
MODE128
PAGE=&6000
CH."$.menu"
```

```
  10 REM PROGRAM ADFS MENU
  20 REM Version B 0.6
  30 REM Author  P.L.Rochford
  40 REM BEEBUG  Aug/Sept 1986
  50 REM Program subject to copyright
  60 :
 100 ONERROR GOTO 910
 110 *KEY9 *EXEC :0.$.!BOOT  M
 120 MODE128:VDU23,1 :DIMA$(47),S%18,C%
18,B%700:R=80:X=140:Y=865:LS=65:LM=60:LH
=45:oldts=VAL(MID$(TIME$,23,2))
 130 TUBE=FALSE:A%=&EA:X%=0:Y%=&FF:IFUS
R(&FFF4)AND&FF00THENTUBE=TRUE
 140 *FX4,1
 150 VDU19,1,0 :PROCN:PROCB(294,30,1250
,829):PROCV(294,30,1250,829):PROCB(420,8
80,1130,975):PROCB(40,562,253,647):PROCB
(40,402,253,487):PROCB(40,242,253,327):P
ROCB(40,80,253,167):PROCB(40,720,250,955
):PROCV(40,720,250,955):PROCface
 160 REPEAT:PROCH:REPEAT:PROCt:F%=INKEY
(0):IFF%>47ANDF%<50OSCLI"MOUNT "+STR$(F%
-48):PROCH
 170 IFF%=136OSCLI"BACK":PROCH
 180 IFF%=139OSCLI"DIR $":PROCH
 190 IF F%=42 PROCstar:PROCH
 200 UNTIL F%>64ANDF%<H%+65
 210 PROCD(A$(F%-64))
 220 IFA%=20SCLI"DIR "+A$(F%-64)
 230 IFC%?3=&80:PROCI
```

```
  240 UNTIL A%<>2 AND C%?3<>&80
  250 IFC%?7<>&80THEN280
  260 MODE7:PROCK:IFTUBE=TRUE THENPAGE=&
800:CHAINA$(F%-64):END
  270 PAGE=&E00:CHAINA$(F%-64):END
  280 MODE7:PROCK:IFTUBE=TRUE THENPAGE=&
800:OSCLIA$(F%-64):END
  290 PAGE=&E00:OSCLIA$(F%-64):END
  300 :
  310 DEFPROCH:oldsx=X:oldsy=Y:oldmx=X:o
ldmy=Y:oldhx=X:oldhy=Y:K%=B%:PROCF:PROCA
:PROCG:PROCJ:PROCQ:VDU19,1,7 :*FX21,0
  320 ENDPROC
  330 :
  340 DEFPROCF:PROCC(8):G%=47-C%?5:IFG%=
0 ENDPROC
  350 H%=0:FORE%=1TOG%:P$="":FORD%=K%+1T
OK%+10:P$=P$+CHR$?D%:NEXT:K%=K%+11:PROCD
(P$):IF((C%?&E EOR2)AND3)=3 H%=H%+1:A$(H
%)=P$
  360 NEXT:ENDPROC
  380 DEFPROCA:PROCC(5):C$="":FORD%=K%+1
TOK%+K%?0:C$=C$+CHR$?D%+CHR$32:NEXT:I%=K
%?(2+K%?0):PROCC(6):D$="":FORD%=K%+3TOK%
+12:D$=D$+CHR$?D%:NEXT:PROCC(7):H$="":FO
RD%=K%+3TOK%+12:H$=H$+CHR$?D%:NEXT:ENDPR
OC
  390 :
  400 DEFPROCG:A%=&71:X%=K%MOD256:Y%=K%D
IV256:CALL&FFF1:J%=K%!0:ENDPROC
  410 :
  420 DEFPROCJ:COLOUR0:COLOUR129:PROCV(4
20,880,1130,975):PROCL(C$,48-LENC$/2,2):
PROCV(40,562,253,647):PRINTTAB(4,12)"Dri
ve":IFI%=0PRINTTAB(4,13)"Zero"ELSEPRINTT
AB(4,13)"One"
  430 PROCV(40,402,253,487):PRINTTAB(4,1
7)"Directory"TAB(4,18);D$:PROCV(40,242,2
53,327):PRINTTAB(4,22)"Library"TAB(4,23)
;H$:PROCV(40,80,253,167):PRINTTAB(4,27)"
Bytes Free"TAB(4,28);J%:ENDPROC
  440 :
  450 DEFPROCQ:VDU28,19,29,75,7:CLS
  460 IFG%=0ORH%=0PRINTTAB(23,6)"NO MENU
FILES!"TAB(9,11)"Press "CHR$136"- to ret
urn to previous Directory"TAB(12,13)"Pre
ss "CHR$139" to go to root Directory"TAB
(18,15)"Or select another Drive":VDU26:P
ROCclk:PROCm:ENDPROC
  470 FORE%=1TOH%:PRINT;"  ";CHR$(E%+64)
;" ";A$(E%);:IFE%MOD4=0PRINT'
  480 NEXT:VDU26:PROCclk:PROCm:ENDPROC
  490 :
  500 DEFPROCC(A%):C%!1=K%:C%!5=47:C%!9=
0:X%=C%MOD256:Y%=C%DIV256:CALL&FFD1:ENDP
ROC
  510 :
  520 DEFPROCD(Z$):$S%=Z$:?C%=S%MOD256:C
%?1=S%DIV256:A%=5:X%=C%MOD256:Y%=C%DIV25
6:A%=(USR(&FFDD)AND&FF):ENDPROC
  530 :
  540 DEFPROCI:PROCV(420,880,1130,975):P
ROCL("Master Sideways RAM Bank Loader",3
2,2):VDU28,21,29,75,7:CLS:OSCLI"ROMS":PR
INT'"Selected: ";A$(F%-64):PRINT'"Load i
nto which bank (4,5,6 or 7) ? ";:REPEAT:
PROCt:G=INKEY(0):UNTILG>51ANDG<57
  550 G$=STR$(G-48):PRINTG$:OSCLI"SRLOAD
 "+A$(F%-64)+" 8000 "+G$+" Q":OSCLI"INSE
RT "+G$:VDU7,7:PRINT'"Press SPACE to ret
urn to Menu, CTRL-BREAK to exit":REPEATG
=INKEY(0):PROCt:UNTILG=32:VDU26:ENDPROC
  560 :
  570 DEFPROCK:*FX4,0
  580 ENDPROC
  590 :
  600 DEFPROCN:VDU23,2,170,85,170,85,170
,85,170,85:GCOL16,1:MOVE0,0:PLOT101,1280
,1024:ENDPROC
  610 :
  620 DEFPROCB(L%,M%,N%,O%):GCOL0,0:VDU2
5,4,L%;M%;25,101,N%;O%;25,4,L%-5;M%+5;25
,101,N%-5;O%+5;:ENDPROC
  640 DEFPROCV(L%,M%,N%,O%):GCOL0,1:VDU2
5,4,L%-2;M%+8;25,101,N%-8;O%+2;:ENDPROC
  650 :
  660 DEFPROCL(A$,W%,V%):A%=&A:X%=&70:Y%
=0:PRINTTAB(W%,V%);:FORZ%=1TOLENA$:?X%=A
SCMID$(A$,Z%):CALL&FFF1:VDU23,240,X%?1,X
%!1;X%!2;X%!3;X%?4,23,241,X%?5,X%!5;X%!6
;X%!7;X%?8:VDU240,8,10,241,11:NEXT:ENDPR
OC
  670 :
  680 DEFPROCt:TS=VAL(MID$(TIME$,23,2)):
IFTS=oldts ENDPROC
  690 GCOL0,1:MOVEX,Y:PLOT6,oldsx,oldsy:
IF TS=0:PROCm
  700 QS=TS*PI/30:GCOL0,0:MOVEX,Y:SX=X+L
S*SIN(QS):SY=Y+LS*COS(QS):PLOT6,SX,SY:ol
dsx=SX:oldsy=SY:oldts=TS:ENDPROC
  710 :
  720 DEFPROCm:MOVEX,Y:GCOL0,1:DRAWoldmx
,oldmy:TM=VAL(MID$(TIME$,20,2)):QM=TM*PI
/30:GCOL0,0:MOVEX,Y:MX=X+LM*SIN(QM):MY=Y
+LM*COS(QM):DRAWMX,MY:oldmx=MX:oldmy=MY:
PROCh:ENDPROC
  730 :
  740 DEFPROCh:MOVEX,Y:GCOL0,1:DRAWoldhx
,oldhy:TH=VAL(MID$(TIME$,17,2))+TM/60:IF
TH=0 PRINTTAB(4,8)LEFT$(TIME$,10)
  750 IFTH>12THENQH=(TH-12)*PI/6ELSEQH=T
H*PI/6
  810 MOVEX,Y:GCOL0,0:HX=X+LH*SIN(QH):HY
=Y+LH*COS(QH):DRAWHX,HY:oldhx=HX:oldhy=H
Y:ENDPROC
  820 :
  830 DEFPROCclk:MOVEX,Y:PLOT157,X+68,Y:
PRINTTAB(4,8)LEFT$(TIME$,10):ENDPROC
  840 :
```

→28

INSTANT TEXTURING

Experiments with &358

In these heady days of mice and icons, the subtle textured background has achieved a certain vogue. David Graham finds a short-cut for the fashion conscious.

There are many ways to create subtle background textures on the Beeb. The one that I want to look at here is of quite stunning economy and simplicity. The technique revolves around a single location in page 3 of the Beeb's memory: location &358 in fact. This holds the byte which is written all over the screen when any CLS (clear screen) instruction is executed. This byte is used for clearing the screen in all modes, though its effect is very different in mode 7 from that in the other modes. In mode 7 it actually fills the screen with characters corresponding to the byte held in &358 (normally 32 in mode 7), and we will look at this first before moving on to its more useful application in the graphics modes.

MODE 7

To test out the effect in mode 7, try the following:

```
MODE 7
?&358=42
CLS
```

The screen will instantly fill with asterisks, because ASCII(42) is the asterisk character. If you want to try other keyboard characters, you can use the following. When this program is run, pressing any key will cause the screen to fill instantly with the chosen character.

```
10 MODE 7
20 A%=GET
30 ?&358=A%
40 CLS
50 GOTO 20
```

This technique of screen filling can be used in conjunction with windows. A text window should be defined using VDU26 before the CLS is executed, and then only the window will be filled with the chosen character. But all this is of relatively limited application. We must turn to other modes to make maximum use of &358.

THE GRAPHICS MODES

In modes 0 to 6, the contents of &358 are written to the screen in exactly the same way as in mode 7. But the effect is very different. Instead of writing a given character to the screen defined by the contents of &358, this byte now determines the colour of each pixel on the cleared screen (a pixel is the smallest discrete unit of colour). Pixel mapping is rather a complex affair that is different in different modes. But we need not get too bogged down here. Some simple examples will readily illustrate the effects.

Perhaps the best mode to start off with is mode 4. To see the effect in action, try the following program:

```
10 MODE 4
20 INPUT A%
30 ?&358=A%
40 CLS
50 GOTO 20
```

Run the program, and enter any number between 0 and 255. The result will be vertical white stripes of varying spacing and density - 255 gives total whiteness.

To see what is going on, enter first 128, then try 64, then 32, then 16, 8, 4, 2 and finally 1. You will see a set of vertical white stripes which shift minutely to the right as you progress down the scale. In other words, each bit of the byte at &358 turns on or off one bit of the background white. When you clear the screen, the byte is written successively into all screen locations, so building up a series of vertical lines.

To produce a vertical stripe effect, we can turn on alternate pixels. The value to put into &358 is therefore 128 + 32 + 8 + 2, or 170. The visual effect achieved by this will depend very much on the resolution of your monitor. On a low resolution screen you will not notice the striped effect at all, and will see a uniform grey. On medium and high resolution screens, the vertical stripe pattern will be more obvious. You can accentuate this by altering the value

contained in &358. 146 and 136 are both worth trying. They give a coarser effect.

To alter the colours of the fill, you can use VDU19. For example, if you insert:

```
45 VDU19,1,6;0;
```

this will give a cyan and black background. The black of course may also be changed using VDU19. For example, inserting the following:

```
47 VDU19,0,1;0;
```

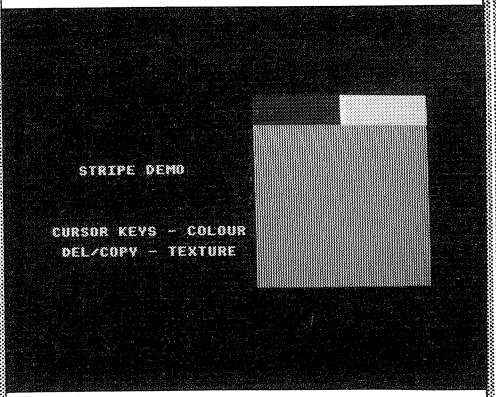
will change it to red, thus mixing cyan and red to give a new colour. The User Guide gives full details on the use of VDU19 to alter physical colours, and you may like to experiment with this.

OTHER MODES

The other modes work in a broadly similar way to mode 4, except that with the multi-colour modes, each bit in the byte at &358 not only determines whether a certain pixel will be on or off, it also determines its colour. Thus for example if you run the program listed above, but with MODE 4 changed to MODE 1, you will see that values of 128, 64, 32 and 16 each give a yellow bar shifting to the right as you descend the scale. But when you get to 8, a red bar appears in the left-most position, and the values 4, 2 and 1 cause the red bar to shift similarly. By adding values together you can get any combination of red and yellow. Thus 90 (64 + 8 + 16 + 2) will give alternate yellow and red stripes. Any value which would 'superimpose' a red over a yellow bar gives a white bar in that position. Thus 136 (128 + 8) gives a single repeated white bar. All four colours produced (black, white, yellow and red) can each be changed to any other using VDU19 again.

In this way it is possible to make quite delicious colour combinations of varying density and shade. Even the most obvious colour combinations - green and red, blue and red, blue and green, or yellow and red produce the most agreeable effects.

The program listed below allows you to sample some of these with relative ease. It works in mode 1, and displays a large window containing the texturing effect. Above this are two small windows which give the two colours currently used in the mix. The cursor keys change the two component colours, and Delete and Copy alter the shading pattern and



density. It is hoped that some of these combinations inspire the reader to experiment with location &358 for himself.

You will find mode 2 worth a look. And there are three adjacent locations that may also be found rewarding.

```
&357||Foreground colour (text)
&359||Foreground colour (graphics)
&35A||Background colour (graphics)
```

Happy hunting!

```
10 MODE1:PRINTTAB(3,9)"STRIPE DEMO"
20 PRINTTAB(0,15)"CURSOR KEYS - COLOU
R"
30 PRINT'" DEL/COPY - TEXTURE"
40 VDU23,1,0;0;0;0;
50 VDU28,21,5,29,2:?&358=15:CLS
60 VDU28,30,5,38,2:?&358=240:CLS
70 VDU28,21,20,38,5
80 A=1:B=2:C=0:PROCSTRIPE:*FX4,1
90 PROCKEY:PROCSTRIPE
100 GOTO90
110 DEFPROCKEY
120 *FX15,0
130 Z=GET
140 IFZ=136 THEN A=A-1:IFA<0 THEN A=7
150 IFZ=137 THEN A=A+1:IFA>7 THEN A=0
160 IFZ=139 THEN B=B+1:IFB>7 THEN B=0
170 IFZ=138 THEN B=B-1:IFB<0 THEN B=7
180 IFZ=127 THEN C=C-1:IFC<0 THEN C=5
190 IFZ=135 THEN C=C+1:IFC>5 THEN C=0
200 ENDPROC
210 DEFPROCSTRIPE
220 VDU19,1,A;0;
230 VDU19,2,B;0;
```

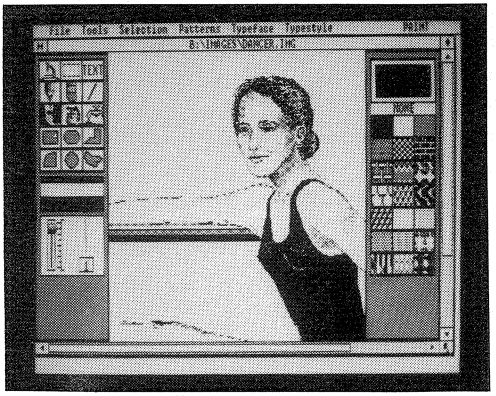
→39

←8

CONCLUSION

This has probably been the most difficult review I have ever had to do on Beeb hardware. I am used to using MS-DOS and IBMs at work, but having a machine in front of you that looks like a Beeb but does not respond or operate like one, can be hard to get used to.

Taking the hardware first, it must be said that Acorn have done a brilliant job in producing a 16 bit co-processor that fits inside the Master on a board the same size as the Turbo. During the review, the unit performed faultlessly.



The bundled software leaves me with mixed feelings. I found GEM Desktop a valuable addition to the system but GEM Paint, although fun to use and quite capable, does not have the power or polish of the excellent AMX Pagemaker which can do all that GEM Paint will do and much more. GEM Write I felt reasonably comfortable with, and like the rest of GEM is very user-friendly. The ability to mix text and images is a nice feature and so is the novelty of using the mouse to select various functions. I found it rather slow in operation like the rest of GEM, though, and felt hampered by the fact that so few pages could be held in RAM, and that text had to be printed from disc via an extra application.

However, the GEM collection is certainly worth having though, and those who buy the Master 512 will be pleased to get it as part of the package. But, the machine is worthy and capable of running much more powerful software.

The extra, and largely unsupported, utilities on the boot disc supplied with the 512 will be of most interest to those who want to experiment with the machine. There are some intriguing ones included, not all of which I have had space to mention. They all add to the power of the machine and deserve investigation by those who wish to go deeper into DOS Plus.

Documentation supplied with the Master 512 I found somewhat lacking. The GEM manual is generally fine but that for DOS Plus I feel is inadequate even for those who want to use DOS at a fairly simple level.

The Master 512 is aimed primarily at the small business and educational markets where the major requirement is the ability to run standard business software packages (no Basic interpreter is supplied). A machine that can run CPM/86 and MS-DOS applications in so many disc formats is a powerful tool when you consider that you have all the strengths of the BBC computer as well. Couple that with the ability to run a large percentage of IBM material and a price tag of just £399 on top of the basic Master 128, and you have some very good reasons for investing in a 512.

←25

The Master Series - An ADFS Menu (contd.)

```
  850 DEFPROCface:GCOL0,0:FORT=0TO2*PI S
TEPPI/6:MOVEX+R*SIN(T),Y+R*COS(T):DRAWX+
(R-4)*SIN(T),Y+(R-4)*COS(T):NEXT:MOVEX,Y
:PLOT149,R-27,Y:ENDPROC
  860 :
  870 DEFPROCstar:PROCV(420,880,1130,975
):PROCL("** S T A R  C O M M A N D **",
34,2):VDU28,21,29,75,7:CLS:VDU23,1,1 :IN
PUT"*"star$:OSCLI(star$):PROCW:ENDPROC
  880 :
  890 DEFPROCW:PRINT'"Press Space";:REPE
AT:PROCt:G=INKEY(0):PROCt:UNTILG=32:VDU2
6:VDU23,1 :ENDPROC
  900 :
  910 IF ERR=17 MODE7:PROCK:REPORT:PRINT
'"Page is currently at &";~PAGE:END
  920 REPORT:PRINT" at line ";ERL:PROCW:
RUN
```

BEEBUG WORDPROCESSING OFFERS

INTERWORD

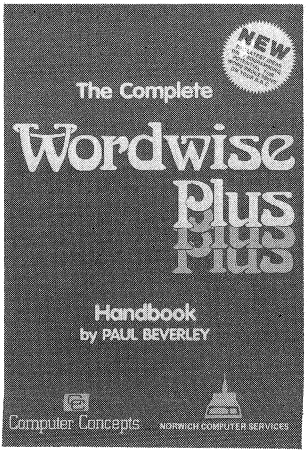
The new wordprocessing Rom from Computer Concepts. Fully integrates with the other programs in their Rom Link suite.

The ROM-LINK suite, of which INTER-WORD is an important part, is a uniquely flexible set of integrated ROMs, compatible with all models of the BBC micro.

Normal Price £56.35
Beebug Price £47.00 plus £1.00 p&p
Wordwise Plus owners may quote their registration number to claim a further £3.00 discount.

The Complete Wordwise Plus Handbook
By Paul Beverley, Norwich Computer Services.

An excellent handbook for anyone using Wordwise Plus. Over 400 pages of information, advice, hints, tips etc etc.



As reviewed in Beebug Aug/Sept. 1986

We are able to offer the book and accompanying disc for the price of £19.00 plus £1.50 p&p. Normal price £25.00 inc p&p.

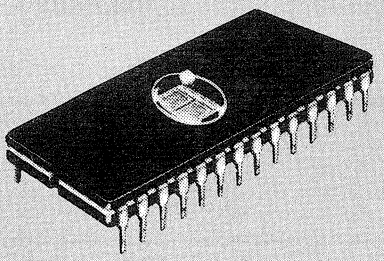
Available from Beebug Retail, Dolphin Place, Holywell Hill, St. Albans, Herts. AL1 1EX.

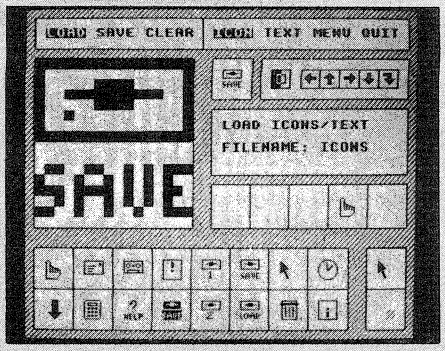
ICON MASTER FROM BEEBUGSOFT

Icons are graphical representations of functions. More powerful mini and micro computers use icons in place of the more traditional menus as a front end for many of their programs.

For example, an icon of a dustbin could be used to represent the *DELETE command, or a clock to ask for the time. Icon Master enables you to add the power of icons to your own programs. Replace your menu screens with dynamic screens of icons, where selection is controlled from the cursor keys. Icons are sophisticated but simple to use as anyone who has [illegible] Mouse will kno[illegible]

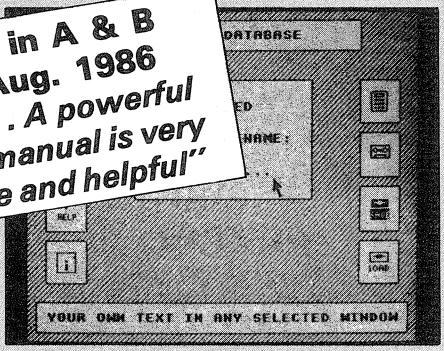
Icon Master is n[illegible] from the keybo[illegible] Mouse owners n[illegible] control it with the[illegible] Icon Master auto[illegible] which you are using. Once you have created your icon front end, you may use it with or without Icon Master in your computer and so may pass it to friends or even include it in software that you intend to sell.





Icon Master is easy to use. Firstly, you create your own Icons on the (icon controlled) designer screen.

As Reviewed in A & B Computing Aug. 1986
"Works well . . . A powerful utility . . . The manual is very comprehensive and helpful"



Then you create your own screen and position your icons anywhere on it. Once you have finished, Icon Master creates the necessary code and adds it to your program for you. This screen shows how you might use icons in a database.

BEEBUG SOFT

16K Rom £34.00 Members £25.50

ANTI-GLARE SCREENS

POWERSCREENS

Eliminates electrostatic charge as well as reflection and glare.

4 sizes — 250mm x 170mm, 300mm × 230mm, 320mm × 260mm, 350mm × 250mm — **£35.59**

RIBBONS

Epson FX80	**£4.50**	Oki 84	**£4.20**
Epson LX 80	**£4.50**	Taxan/Kaga	**£4.50**

FLOPPY DISCS

5¼" lifetime warranty (boxes of ten)

	1-4	5-9	10+
SS SD 48tpi	**£13.20**	**£12.65**	**£12.00**
SS DD 48tpi	**£13.80**	**£13.20**	**£12.55**
DS DD 48tpi	**£17.82**	**£17.25**	**£16.00**
DS DD 96tpi	**£20.70**	**£19.50**	**£18.30**

5¼" Storage — FF15 Flip 'n' File for 15 discs **£7.00**

STATIONERY

11 × 9½ 1pt listing paper micro perfs (2000 sheets) **£13.50**

All prices include VAT and carriage. Free comprehensive catalogue upon request.
Access & Barclaycard accepted.

Advanced Media Limited, Media House, 6 Knolls Way, Clifton, Beds. SG17 5QZ.
Tel: 0462 811817. Telex: 825644.

MIRRORSOFT BY MAIL ORDER

The entire Mirrorsoft range of quality software is available by mail order direct from our warehouse, with no extra charge for postage and packing (UK only). All programs run on BBC B/B+ and Electron. Please enquire about Master-compatible versions.

Prices given are for tape versions. Please add £3 for 5.25" disks. Please state whether 40 or 80 track required.

We accept cheques, postal orders, Access, VISA, and American Express. Please allow 28 days for delivery.

Send for our free colour catalogue, too!

EARLY LEARNING/CHILDREN'S ENTERTAINMENT

Caesar's Travels	£12.95(disk only)	Ages 3-9
Count with Oliver	£7.95	Ages 4-7
Crack It! Towers	£9.95	Age 7+
First Steps with the Mr Men	£8.95	Ages 3-6
Giddy Game Show	£9.95	Ages 3-6
Here & There with the Mr Men	£7.95	Ages 4-7
Look Sharp!	£7.95	Ages 5-11
Mr Men Magic Storymaker	£9.95	Ages 4-7
Quick Thinking!	£6.95	Ages 5-11
Quick Thinking Plus!	£6.95	Ages 5-11
Word Games with the Mr Men	£9.95	Ages 5-8

CREATIVE COMPUTING

Fleet Street Editor	£39.95 (disk only)
Admin Xtra	£14.95 (disk only)
Fonts 'n' Graphics	£14.95 (disk only)
Walt Disney Graphics	£14.95 (disk only)

GAMES

Caesar the Cat	£6.95
Hi Bouncer	£6.95
Strike Force Harrier	£9.95

HOME DISCOVERY

BBC Mastermind	£9.95
BBC Mastermind Quizmaster	£5.95
Joffe Plan	£9.95
Know Your Own Psi Q	£9.95
Know Your Own Personality	£9.95
Star Seeker	£9.95

MIRRORSOFT
Dept BB,
FREEPOST BS4382,
Paulton, Bristol BS18 5BR (No stamp needed)

Tube Compatibility of ROM Software

We reproduce below the latest updates and additions to the ROM list compiled by Benjamin Rietti for Tubelink (Prestel page *258216#), showing the compatibility of ROMs with the Tube and Second Processor. The full list was reproduced in the supplement in BEEBUG Vol.4 Nos. 7 and 8 and last updated in Vol.5 No.1. Please notify us (or Benjamin) of any errors or omissions. We endeavour to ensure that the information published is accurate but we cannot be held responsible for any errors how-so-ever caused.

Compatibility Coding:

'C' = Completely Compatible over Tube
'*' = Memory access is from I/O proc.
'**'= Allows both memory sets
'P' = Partly Compatible
'I' = Incompatible

Rom Name	Code C,P,I	Company
ADE 2	C**	SYSTEM
ADV.ROM MANAGER	C*	A.C.P
ADV.1770 DFS	C	A.C.P
BEEBAID	C	JAYSOFT
BITSTIK I	C	ROBOCOM
BITSTIK II	C	ROBOCOM
BROM PLUS	P	CLARES
DUMPMASTER II	C	BEEBUGSOFT
DUMPOUT III	C	WATFORD ELECTRONICS
IMAGIN.A	I	IBBOTSONS
INDEX UTILITY	C	DATASTORE
INTERWORD	I	COMPUTER CONCEPTS
INTERBASE	I	COMPUTER CONCEPTS
MACRO ASSEMBLER ROM	C**	STAR SOFTWARE
MATRIX ROM	C	VINE MICROS
MICRON PROGRAMMER	C	H.C.R
PAGEMAKER PLUS	I	A.M.S
RAMROD	C**	CLARES
SOFTLINK CONTROLLER	C	ABACUS
WORDEASE	C	BEEBUGSOFT

This list and the information contained within it, remains the copyright of Benjamin Rietti (Editor of Viewfax Tubelink). Any reader wishing to know more about Tubelink's Postal Service scheme should send an S.A.E. to Tubelink at:

P.O. BOX 641, LONDON NW9 8TF.

Events

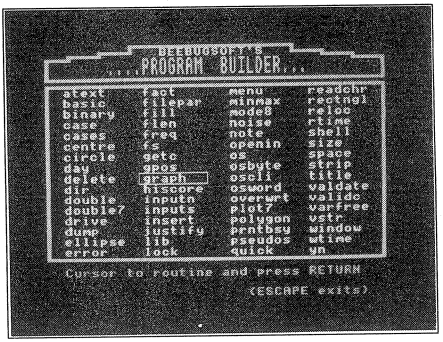
PCW Show	Olympia, London	3- 7 Sept
Electron and BBC Micro User	UMIST, Manchester	26-28 Sept
Electron and BBC Micro User	Royal Horticultural Halls, Westminster	7- 9 Nov

PROGRAM BUILDER — 60 handy basic routines

Program Builder brings you over sixty complete routines ready to use when writing your own Basic programs. Why waste time writing frequently used code when you can rely on professional routines written for you, that will work first time.

Program Builder offers over 60 functions and procedures which may be combined with your own Basic program. They are all fully documented and use standard variables to ensure that they will be compatible with your work.

A wide range of topics are covered by Program Builder, which is fully automated. Just select the desired routine from an on-screen menu and the code will be automatically added to your program.



Any number of routines may be put together in a single program.

There is not the space to list all the routines in Program Builder but here is a small selection to give you an idea of its range:

- Input & validate date
- Binary from decimal
- Menu & return input
- Special sound effects
- Strip spaces
- Text any size/height/angle
- Double height text any mode
- Graphics plotting
- Circle plotter
- String entry/justify/insert
- Plot ellipse
- Quicksort
- Screen dump
- Password
- Mode 7 graphics
- Yes/No input
- Text justify
- Print string vertically
- Error handling
- A new mode ... Mode 8

Members Price: Tape £12.00 — Disc £15.75

PROGRAMMERS

Now is the time to contact us with your ideas and programs for inclusion in the Beebugsoft range.

All submissions treated in strictest confidence.

Make money from your hobby.

Contact:
The Software Manager
Beebugsoft
P.O. Box 50
St. Albans,
Herts.

Classified Ads

As from the October issue of BEEBUG, we are offering all BEEBUG members the opportunity to place personal ads free of charge in the magazine (see Editorial Jottings). Please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any ad if necessary. We shall still continue to accept members' business ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately in future. Please send all ads (personal and business) to MEMBERS' ADS, BEEBUG, Dolphin Place, Holywell Hill, St. Albans, Herts. AL1 1EX. The normal copy date for receipt of all ads will be the 12th of each month.

HEAD-MASTER ROM: Prints your own headed paper with Kaga/Canon or Epson compatible printer. Any size/shape headings, with graphics and different typefaces, to your (or our) design, on ROM, called with a simple *command. Prices from £25. For details and samples send A5 SAE to HEAD-MASTER, 206 Barker Drive, St.Pancras Way, London.

Problems? try R-SOFT utilities!
1. HOW-TO: An essential collection of software and instructions for frustrated new disc owners who want to move their programs to disc.
3. ROMPULL + TAPEDUMP.
4. AUTOMATIC DISC MENU: Includes sideways-RAM version, can boot from ROM, works with 2nd 6502.
6. SWROM*: Puts your Basic / machine code programs in ROM format.
7. RFS-GENERATOR: Generates ROMs for the *ROM filing system. This does not use the DFS workspace and is an ideal tool to run nasty programs from disc.
All above packages £5.00 each.
13. D-MASTER-V2: Superb disc backup program; needs 8271 and swram; £8.00; on 16K EPROM £12.00. Upgrade for D-MASTER owners: £500/£9.00.
16. OPUS SD/DD/Challenger: Disc menu, disc editor, disc indexer, etc. Please enquire for further details.
17. D-EDITOR: based on D-MASTER-V2: will edit any disc D-MASTER-V2 can backup! £8.00/£12.00.
Please send SAE for full list.
R-SOFT, 22 Marriotts Close, Felmersham, Bedford, MK43 7HD. (0234) 781730.

Music 500, issues 1 and 2 tapes + 1 year Ample User subscription £100 ono. Ikon Ultradrive 1.04 UFS + manual and 18 mini cassettes £80. M.Lee (0535) 603277.

For Sale: Wordstar Professional suite for Z80 £225, Acornsoft GXR £18, Ultracalc 2 £30, Exmon 2 £15, DNFS 1.2 £10, Basic 2 £10. Acorn Prestel Adaptor £70. All open to offers. Phone (0533) 312661.

BBC with Torch Z80, twin 800k drives, Hi-res Kaga green monitor, cassette recorder, double plinth, ATPL board with 16K sideways RAM. View, Viewsheet. All perfect with manuals plus Wordstar Professional, DBaseII, Supercalc 2. All fully configured for Torch/BBC. Includes 'Perfect' software with Torch processor. All software is original with manuals. Nothing pirated at all. Complete system for business user, club secretary etc. £1250 the lot. 01-552 1467 (evenings).

BEEBUG back issues (bound) Vol.1 No.1 to Vol.4 No.8. Offers: R.Haynes (office) 01-729 3299, (Home) 01-672 8376.

4 BBC-Bs, 100K disc drives, monochrome monitors, Edword wordprocessor ROMs. £400 each ono. London School of Foreign Trade, 89 Westminster Bridge Road, London SE1. 01-928 6810.

STARdataBASE. Original, with registered number. Improved Utilities. £30. Tel:(0705) 592489.

BBC B+ with speech ROMs, View, Basic Editor £350. Z80 2nd processor with all software £270. Tel: (0225) 743710 near Bath.

Sideways RAM Board 16K with disc software £15. (0903) 892682.

BBC B+128, second processor, View 3, Viewsheet, ADFS £475 ono. Chris, 5 Devonshire Road, Westbury Park, Bristol. Tel:(0272) 45690.

→ 36

DUMPMASTER — NOW IN ROM

The extremely versatile Dumpmaster program is now available on 16K Rom and supports more printers and offers even more features.

Dumpmaster provides fast machine code dumps using up to 8 shades and will accurately copy your screen in any of the graphics modes and even in the teletext mode. It also includes a special 'snapshot' facility to produce screen dumps from a game, or any other program, at the press of a key (unless protected).

The tape and disc versions create tailor-made dumps which may be called as required or even appended to your own Basic program. The Rom version has the added advantage that it uses up no memory and is always available from a single command. A number of additional features are available on the Rom version, including printing of windows, text dumps and vertical dumps (printer dependent).

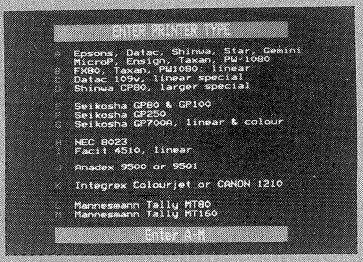
PRINTERS SUPPORTED

Anadex DP 9000 series
*Brother HR5
*Canon PJ1080A
Cosmos 80
Ensign 1650
Epson FX, LX
Gemini
Integrex 132 Colour
Mannesmann Tally MT80
NEC PC 8023
*Olivetti JP101
*Quen Data DP100
Seikosha GP100
*Seikosha GP550
Shinwa CTI CP80
Star all
*Tandy DMP-100
Brother M1009
Canon A-1210
Canon PW series
Datac 109V
Epson all
Facit 4510
*IDS-480
Kaga Taxan
Mannesmann Tally MT160
Micro-P MP-165
Panasonic KX-P series
Seikosha GP80
Seikosha GP250
Seikosha GP700A
Star DP-8480
*Tandy CGP115

*Only available on Rom version

UPGRADE

A trade-in discount of 50% of the disc/tape price is offered when upgrading to Rom.



Members Price: Tape £9.00 — Disc £11.25 — Rom £25.50.

CLASSIFIED ADS (continued)

DIGIT: Image Analysis for BBC and Grafpad. Count objects, measure distances, thicknesses. Digitise shapes, measure perimiter, area, radius, volume, Feret diameters, C of G, shape factors. Edit, display and print results as histogram with mean and S.D. Uses: Quantatative Microscopy, Map Measurement, Metallurgy etc. 40 or 80 track disc and instruction manual. £70. For GrafpadII (requires Grafpad utilities disc) or A3 Grafpad. Institute of Opthalmology, Judd St., London, WC1H 9QS. 01-387 9621 ext.64.

6502 Second Processor; Hi-View Disc; Hi-Basic ROM; DNFS ROM, hardly used, complete with manuals, £180. Write: R.Christian, Croit-y-Keeil, Port Grenaugh, Santan, Isle of Man.

BBC B, O.S 1.2, Help ROM, manuals, joysticks, colour monitor, £450 ono. Phone A.Rodgers 01-446 3033 evenings.

ELECTRON complete with £70 of BEEBUGSOFT software £50. R.Barnes, Tel:(05827) 5929. Also Sinclair QL with preliminary Basic £50 ono.

Members' Corner

BEEBUG member John Childs of Huntingdon has had the following items of hardware and software stolen from his home. Anyone being offered any of these or knowing of their whereabouts should contact St Neots police Station on 0484-73131.

HARDWARE	SOFTWARE	
Shinwa Printer	Acornsoft/Mirle Business Applications	Teletext
Ferguson data Recorder	ISO-Pascal	Paintbox
Robin Touch Light Pen	View Index	Magscan
Solidisc 64K Sideways RAM	View Printer Driver	(BEEBUGSOFT)
	Database	
	(ACORN)	

New High Scores

BEEBUG

Supplier	Game	Score	Player
ACORNSOFT	Labyrinth	630000	P.Allen
ACORNSOFT	Volcano	6150	R.F.Hagart
ADDICTIVE	Boffin	253435	B.Jackson
BEEBUG GAMES	Santa	51720	S.Williams
BEEBUG MAG	Scrumpy	45165	D.Callaghan
FIREBIRD	Duck	64250	J.Greig
FIREBIRD	Hacker	13480	O.Upton
FIREBIRD	Microcosm	4087	O.Upton
SUPERIOR SOFT	Repton	660645	R.F.Hagart

LOW COST C.A.D.

ATTENTION ALL ELECTRONICS CIRCUIT DESIGNERS!!

IBM PC (and compatibles) BBC MODEL B, B+ and MASTER, AMSTRAD CPC and SPECTRUM 48K

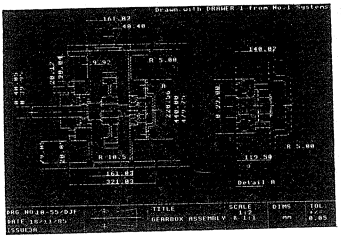
ANALYSER I and II compute the A.C. FREQUENCY RESPONSE of linear (analogue) circuits. GAIN and PHASE, INPUT IMPEDANCE, OUTPUT IMPEDANCE and GROUP DELAY (except Spectrum version) are calculated over any frequency range required. The programs are in use regularly for frequencies between 0.1Hz to 1.2GHz. The effects on performance of MODIFICATIONS to both circuit and component values can be speedily evaluated. Circuits containing any combination of RESISTORS, CAPACITORS, INDUCTORS, TRANSFORMERS, BIPOLAR AND FIELD EFFECT TRANSISTORS and OPERATIONAL AMPLIFIERS can be simulated — up to 60 nodes and 180 components (IBM version).
Ideal for the analysis of ACTIVE and PASSIVE FILTER CIRCUITS, AUDIO AMPLIFIERS, LOUDSPEAKER CROSS-OVER NETWORKS, WIDE-BAND AMPLIFIERS, TUNED R.F. AMPLIFIERS, AERIAL MATCHING NETWORKS, TV I.F. and CHROMA FILTER CIRCUITS. LINEAR INTEGRATED CIRCUITS etc. STABILITY CRITERIA AND OSCILLATOR CIRCUITS can be evaluated by "breaking the loop".
Tabular output on Analyser I. Full graphical output, increased circuit size and active component library facilities on Analyser II.
Check out your new designs in minutes rather than days.
ANALYSER can greatly reduce or even eliminate the need to breadboard new designs.
Used by INDUSTRIAL GOVERNMENT and UNIVERSITY R & D DEPARTMENTS worldwide. IDEAL FOR TRAINING COURSES. VERY EASY TO USE. Prices from £20 to £195.

LOW COST COMPUTER DRAUGHTING

On the BBC MODEL B, B+

DRAWER I enables drawings to be created and modified, quickly, easily and with the minimum of hardware. All of the major program elements are written in machine code giving exceptional speed of operation.

FEATURES
- Rubber Banding for drawing lines.
- Solid or Dotted line types.
- Circles, Arcs and partial or complete Ellipses.
- Vertical or Horizontal Text.
- Pan and Zoom.
- Merging of drawings and library symbols from disc.
- Up to 20,000 lines on a drawing.
- • Snap to a user defined grid.
- Absolute or Relative cursor co-ordinates displayed on screen.
- Input from analogue joystick, mouse or trackerball.
- Output to standard dot matrix printer.
- Price from £45 excl. VAT.



Minimum Hardware Required: BBC Model B. Single or Dual 5.25" Disc Drive — 40 or 80 track. T.V. or monitor. Games joystick, Mouse or Trackerball. Dot matrix Printer (Epson 80 series or Epson compatible — BBC Default Mode.

Full AFTER SALES SERVICE with TELEPHONE QUERY HOT LINE and FREE update service.

For illustrated leaflets and ordering information please write or phone

NUMBER ONE SYSTEMS LIMITED
Ref: BB Crown Street, St. Ives, Huntingdon,
Cambs PE17 4EB. Tel: 0480 61778.

ADVERTISING IN BEEBUG

For advertising details please contact:

Lynn Lloyd on (0923) 677222 or Yolanda Turuelo on (0727) 40303

or write to
P.O. Box 50
St. Albans Herts

```
 2530 IF lock?n=-(s$="") THEN PROCos("AC
CESS :"+D$+"."+name$(n)+s$):lock?n=ABSNO
T(s$="")
 2540 ENDPROC
 2550 :
 2560 DEFPROCdelete(n)
 2570 PROCos("DELETE :"+D$+"."+name$(n))
 2580 ENDPROC
 2590 :
 2600 DEFPROCdump(n)
 2610 PROCos("DUMP :"+D$+"."+name$(n))
 2620 PRINT"Any key to continue":x=GET
 2630 ENDPROC
 2640 :
 2650 DEFPROCrename(n):LOCAL i,a$
 2660 FOR i=0 TO 8:a$=a$+CHR$(new?(9*n+i
)):NEXT
 2670 PROCos("RENAME :"+D$+"."+name$(n)+
" "+a$):name$(n)=a$:ENDPROC
 2680 :
 2690 DEFPROCcompact
 2700 CLS:PROChwm:$&380="COMPACT "+D$
 2710 *K.0X%=&80:Y%=&3:CA.&FFF7|MCH."DIS
CMAN"|M
 2720 *FX138,0,128
 2730 PROCstop(0)
 2740
 2750 DEFPROCcopy(p)
 2760 PROCos("COPY "+D$+" "+CHR$cmd?p+"
"+name$(p)):ENDPROC
 2770 :
 2780 DEFPROCos(c$):LOCALX%,Y%
 2790 PRINT"*"+c$:X%=&80:Y%=3:$&380=c$:C
ALL&FFF7:ENDPROC
 2800 :
 2810 DEFPROCexec(a$,n$)
 2820 $&380=a$+" :"+D$+"."+n$
 2830 *KEY0X%=&80:Y%=3:CALL&FFF7|M
 2840 *FX138,0,128
 2850 PROCstop(0)
 2860 :
 2870 DEFPROChwm:LOCAL Y%
 2880 A%=180:X%=PAGE DIV 256:CALL-12
 2890 ENDPROC
 2900 :
 2910 DEFPROChex(c,!&70,?&74)
 2920 VDUc : CALL code:ENDPROC
 2930 :
 2940 DEFPROCasmbl
 2950 FOR p=0 TO 2 STEP 2:P%=code
 2960 [ OPT p
 2970 LDA &74:LSR A:TAX:BCS K:DEX
 2980 .K LDA &70,X:BCS N:PHA:LSR A
 2990 LSR A:LSR A:LSR A:JSR out:PLA
 3000 .N AND #&F:JSR out:CLC:DEX:BPL K
 3010 RTS
 3020 .out TAY:LDA hx,Y:JMP &FFE3
 3030 .hx:]:NEXT:$P%="0123456789ABCDEF"
 3040 ENDPROC
 3050 :
 3060 DEFPROCnewname(n)
 3070 V=9*P?n:VDU31,2,(n-1) MOD 20,129
 3080 FOR r=0 TO 8:new?(V+r)=ASCMID$(nam
e$(P?n),r+1,1):NEXT
 3090 r=0:REPEAT
 3100 IF r=1 THEN r=2:VDU46:GOTO 3100
 3110 x=GET:IF x=127 AND r=0 THEN VDU7:G
OTO 3190
 3120 IF (x=127 OR x=136) AND r=2 r=1:VD
U8
 3130 IF x=127 THEN r=r-1:VDUx:GOTO 3190
 3140 IF x=136 AND r>0 r=r-1:VDU8:GOTO31
90
 3150 IF x=137 AND r<9 r=r+1:VDU9:GOTO31
90
 3160 IF x=13 THEN 3190
 3170 IF r=9 OR x<32 OR x>126 THEN VDU7:
GOTO 3190
 3180 new?(V+r)=x:VDUx:r=r+1
 3190 UNTIL x=13:VDU31,3,(n-1) MOD 20
 3200 FOR r=0 TO 8:VDU new?(V+r):NEXT
 3210 VDU31,38,(n-1) MOD 20:ENDPROC
```

POINTS ARISING POINTS ARISING POINTS ARISING POINTS

PAINTING BY NUMBERS BEEBUG Vol.5 No.3)

We are sorry but the last eight lines of the FILLER program were accidentally omitted from the magazine listing last month. The lines below should be appended to your copy of this program. The magazine cassette/disc version is quite complete and no further action is required in that case. We apologise for any inconvenience.

```
 1210 ]
 1220 mode=zp+40:coltab=mode+8
 1230 NEXT
 1240 FORI%=0TO44STEP4:READI%!zp:NEXT
 1250 DATA &70230F8,&80010102,&30230FC
 1260 DATA &40110201,&10230FE,&A0550400
 1270 DATA &70158F8,&40010101,&30158FC
 1280 DATA &A0110200,&FF100800,&FFFF2018
```

←22

Filer Accounts Option (contd.)

```
6020 IF NOT open% ENDPROC
6040 PRINT"File closed - ";rec-1;" reco
rds in use"
6060 CLOSE#0:open%=0:debit%=0:PROCclear
6080 ENDPROC
6100 :
6300 DEF PROClistc:LOCAL I
6320 PROCwindow1:CLS
6340 FOR I=1 TO N:PRINT com$(I):NEXT I
6360 PROCwindow2:PRINT"Press any key to
continue";
6380 I=GET:PROCclear:PRINT:ENDPROC
6400 :
20000 DEF FNask(msg$)
20020 LOCAL A:PRINT msg$;
20040 REPEAT:A=INSTR("YyNn",GET$):UNTIL
A:IF A>2 THEN PRINT"N" ELSE PRINT"Y"
20060 =A
20080 :
20200 DEF PROCclear
20220 PROCwindow1:CLS
20240 PROCwindow2:ENDPROC
20260 :
20400 DEF PROCwindow1:IF w=2 X=POS:Y=VPO
S
20420 VDU28,0,19,79,3:w=1:ENDPROC
20440 DEF PROCwindow2:VDU28,0,24,79,21
20460 VDU31,X,Y:w=2:ENDPROC
20480 :
20700 DEF PROCread(n,C,R):LOCAL I
20720 PTR#C=FDR+recs*(n-1)
20740 FOR I=1 TO f:INPUT#C,record$(I,R):
NEXT I
20760 ENDPROC
20780 :
20800 DEF PROCwrite(n,C,R):LOCAL I
20820 PTR#C=FDR+recs*(n-1)
20840 FOR I=1 TO f:PRINT#C,record$(I,R):
NEXT I
20860 ENDPROC
20880 :
21200 DEF FNstrip(p$,c$):LOCAL I:I=LENp$
+1
21220 REPEAT:I=I-1:UNTIL MID$(p$,I,1)<>c
$ OR I=0
21240 =LEFT$(p$,I)
21260 :
21600 DEF FNfield(p$):LOCAL I:I=0
21620 REPEAT:I=I+1:UNTIL p$=field$(I) OR
 I>f
21640 =I
21660 :
21800 DEF FNval(p)=VAL(record$(p,0))
21820 :
22000 DEF FNchop(d,n):LOCAL i,m,p:IF n=1
 =1
22020 m=1:REPEAT:m=m*2:UNTIL m>=n+1
22040 m=m/2:i=m
22060 REPEAT:i=i/2
22070 IF m>n REPEAT:m=m-i:i=i/2:UNTIL m<
=n
22080 PROCread(m,F,0):p=FNval(date)
22100 IF d<p THEN m=m-i ELSE IF d>p THEN
 m=m+i
22120 UNTIL d=p OR i=1
22140 REPEAT:PROCread(m,F,0):m=m-1:UNTIL
 FNval(date)<d1 OR m=0
22160 IF m=0 m=1
22180 =m
22200 :
30000 DEF PROCstar(p$):LOCAL I
30010 PROCwindow1:CLS:PROCoscli(p$):PROC
window2
30020 PRINT"Press any key to continue";
30030 I=GET:PROCclear:PRINT
30040 ENDPROC
30050 :
30060 DEF PROCoscli($os)
30070 LOCAL X%,Y%
30080 X%=os:Y%=os DIV 256:CALL &FFF7
30090 ENDPROC
30100 :
31000 DEF PROCerror
31010 IF ERR=17 VDU15:PROCclear:PRINT'"C
ommand aborted":GOTO 140
31020 IF ERR>189 AND ERR<208 REPORT:PRIN
T:GOTO140
31030 ON ERROR OFF
31040 PROCwindow2
31050 REPORT:PRINT" at ";ERL
31060 PROCclose:VDU26:*FX4,0
31070 ENDPROC
```

←27

```
240 IFC=0 THEN D=165
250 IFC=1 THEN D=195
260 IFC=2 THEN D=135
270 IFC=3 THEN D=120
280 IFC=4 THEN D=240
290 IFC=5 THEN D=15
300 ?&358=D:CLS
310 ENDPROC
```

Note: to restore cursor key editing, press Break or use *FX4,0.

DEFAULTS

Contents of &358
MODE 7 = 32
OTHER
MODES = 0

Mode changes
reset default

PRINTER GRAPHICS (part 1)

In the first of three articles, Alex Kang introduces the subject of printer graphics, all readily adaptable to your own printer.

In the first article of this series we will be looking at the way in which graphics are printed on Epson compatible printers. This will lead us on to quite complex graphics applications; and in parts two and three of the series we will develop a flexible screen dump which will print any screen window. In the final part, a complete page-maker will be presented, allowing the easy composition of pages containing graphics and text in any combination.

PRINTER CONTROL CODES

The whole of this series of articles revolves around the direct control of the printer from software run on the BBC micro. All printers have the facility for some kind of software control, and most use what is called an Escape sequence to signify control characters. Thus if you send the letter K to your printer it will (hopefully) print a "K" on the paper, but if that letter is preceded by character 27 (the ASCII Escape character), then your printer will probably behave differently. On an FX80 or similar, it will turn on the graphics printing option, and so on. It is Escape sequences such as this which allow us to control the printer in a completely user-defined manner.

To send control codes to the printer on a BBC micro is relatively simple. You must first turn on the printer with VDU2 (VDU3 turns it off). Then each character you send must be preceded by a VDU1. VDU codes can be strung together, so that the following sequence will turn on the emphasized printing mode on an FX80 or similar:

```
VDU2,1,27,1,69,3
VDU1,27,1,mode%,1,N1%,1,N2%,1,data1,1,
data2,1,data3,1....etc.
```

The code for emphasized printing is Escape E (ASCII 27 and 69). The previous string sends VDU2 to turn on the printer, then 27 and 69, each preceded by a 1; and then an optional 3 to turn the printer off again.

code	ASCII	Description
ESC @	27,64	Reset code
ESC E	27,69	Emphasized ON
ESC F	27,70	Emphasized OFF
ESC G	27,71	Double strike ON
ESC H	27,72	Double strike OFF
ESC 4	27,52	Italics ON
ESC 5	27,53	Italics OFF

A selection of Epson FX80 codes

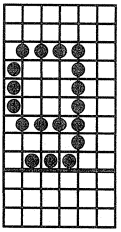
The following short program turns on underlined italics on an FX80 or similar, and prints some text:

```
10 VDU2:REM Turns printer on
20 VDU1,27,1,52,1,27,1,45,1,1
30 PRINT "This is underlined italics"
40 VDU1,27,1,64:REM reset printer
50 VDU3:REM Turns printer off
```

So far we have talked of printer control codes in terms of setting up particular text styles, but in this series we shall be more concerned with directly controlling the movement of the print head, and the printing pins themselves, so that we can achieve screen dumps, and the mixing of text and graphics. We will look first at the vertical positioning of the paper relative to the print head.

The movement of paper between lines is called a line feed, and the distance the paper moves is the line space. The default line spacing on most printers is 1/6 inch, which produces six lines of print per inch. Figure 1 shows a text character set in its design matrix. As may be seen, each dot occupies 1/72 of an inch, so that each default line spacing is equivalent to 12 vertical dots in height.

Figure 1:



Line Spacing

The line spacing may be altered using Escape A n where n represents n/72 of an inch (ASCII equivalent: 27,65,n). The short program below will show the difference in line spacings:

```
10 REPEAT
20 INPUT "Line spacing (8-24):" n%
30 UNTIL n%>=0 AND n%<85
40 VDU2,1,27,1,65,1,n%
50 FOR i%=1TO3
60 PRINT "Line spacing ";n%;"/72 in."
70 NEXT:VDU1,27,1,64,3
```

The Escape sequence 27,64 (or Escape @) is a useful one to remember. It resets the printer to its default settings. It is often a wise precaution to issue this both at the start and the end of a printout.

PRINTER GRAPHICS

In the printer graphics modes, the user program controls exactly where each dot is printed, column by column and line by line. Normally, only eight of the nine pins on the print head are used for graphics, and we can imagine these eight pins as being eight bits in a data byte, with the most significant bit as the top-most pin.

The two most common graphics modes are:

Single density (480 dots/8" line)
set with Escape K (ASCII 27,75)

Double density (960 dots/8" line)
set with Escape L (ASCII 27,76)

The difference between them is simply that in the single density mode the print head moves twice as far horizontally between dot impressions as it does in double density.

These two commands must both be followed by a pair of numbers N1% and N2% which tell the printer how many data bytes to expect. N1% carries the low-byte of the number, and N2% the high-byte. Thus if the number of data bytes to be sent is D%, then:

```
N1%=D% MOD 256
N2%=D% DIV 256
```

Following these are the data bytes themselves. So the format becomes:

```
VDU1,27,1,mode%,1,N1%,1,N2%,1,data1,1,
data2,1,data3,1....etc.
```

where mode%=75 for single density and mode%=76 for double density. Let us take an example. Suppose you define a pattern using just two bytes as illustrated in Figure 2. The values of each byte (or each column) are calculated as shown. We can print 20 sets (ie.40 data bytes) of this pattern with the following:

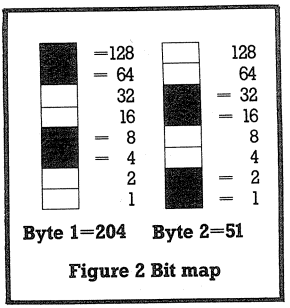


Figure 2 Bit map

```
10 D%=40
20 mode%=75
30 sets%=20
40 N1%=D% MOD 256:N2%=D% DIV 256
50 data1%=204:data2%=51:VDU2
70 VDU1,27,1,mode%,1,N1%,1,N2%
80 FOR set=1 TO sets%
90 VDU1,data1%,1,data2%
100 NEXT
110 VDU1,27,1,64,3
```

To print 5 lines of the above pattern, add the following lines:

```
60 FOR line=1TO5
65 VDU1,27,1,65,1,8,1,13
105 NEXT
```

Try changing mode% to 76 to see the effect of printing in double density. To produce the same width of pattern in double density, you will need to double the values of D% and sets%. If your printer has other graphics modes such as high speed double, or quadruple density, you may like to try them to see their effect.

In the next issue we will develop a flexible screen dump which allows the dumping of any part of a screen in any graphics mode, and will discuss ways in which it may be tailored to the requirements of individual printers.

Wordwise Plus HAND-BOOK

Does the world really need yet another book on Wordwise from the Wizard of Norwich? David Graham conjures up the answer.

THE COMPLETE WORDWISE PLUS HANDBOOK by Paul Beverley, Norwich Computer Services, 31 Cattle Market Street, Norwich NR1 3DY. Tel. 0603-621157. Price: £17.50 inc p. & p. Disc £7.50

If you were at the recent Micro User show, then you probably saw copies of Paul Beverley's latest ouvre chained in their hundreds to his wire-frame carousel. To liberate one would have cost you £15.00 at the time: it now costs £17.50 by post. But is this rather expensive book good value for money?

To begin with it is an exceedingly large book - some 400 pages in length, and done out in Computer Concepts green. Though in spite of using CC's house style, the book is a quite independent production of Beverley's 'Norwich Computer Services'.

The book is organised into 16 chapters with some useful appendices and an excellent index. The last 10 chapters (and in fact 300 of the book's 400 pages) deal with various aspects of the Wordwise Plus programming language, so that very little of the book is of relevance to Old-Wordwise users. The fact that so much of the book is devoted to the programming language should not necessarily deter potential readers however, because much of the treatment is applications-orientated with a multitude of program listings of potentially wide interest. But more of this later.

What of Beverley part one? The first six chapters of this book will be extremely useful to any user of Wordwise or Wordwise Plus. The book is not written for the absolute beginner in the manner of the early chapters of Bruce Smith's book on Wordwise (Wordwise Plus - a user's guide, Collins £9.95 reviewed briefly in Beebug Vol. 4 No. 7), but addresses itself to a number of specific topics. In this context Beverley gives very thorough coverage of the interface between word processor and printer: not the interface in a hardware sense, but the way in which Wordwise may be persuaded to print what you want where you want it.

As ever, Beverley writes clearly, and moreover he writes from the point of view of the day-to-day user of Wordwise: not an unreasonable stance since the book itself was written with it. But the consequence of this is that some of the oddities of Wordwise and Wordwise Plus that you may have been wrestling over for months, are explained and overcome. For instance, if you are using a printer, such as the Epson FX80, that cannot normally print on the top line of the paper, the only effective way to use paging is to set up TS as zero, and BS as around 8 or 10.

Or on another point, have you ever had embedded Wordwise printer commands, particularly at the start of a document, ignored by your printer for inexplicable reasons? Beverley has solved it! The answer is that printer commands which have any form of indent command (IN or TI) between them and the next piece of text, are ignored. So if you are indenting, indent first, and send printer commands afterwards.

The book contains a good number of such observations which will, I am sure, save a great deal of teeth grinding and hair tearing; but it also contains much useful specific advice and general information, as befits a handbook of quality.

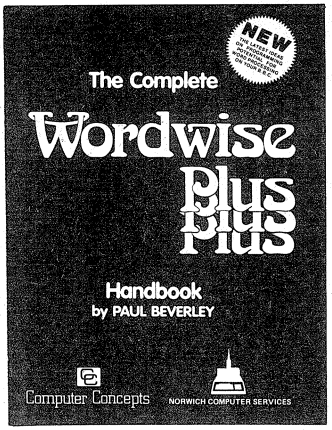
The second, and major part of the book does not fall so readily into the handbook category as the first. It consists largely of some sixty programs written in the Wordwise Plus programming language. These are grouped under six major headings:

Text manipulation
Document layout
File manipulation
Database applications
Numbers and numbering
Calculations

Of these, the Database applications section is arguably the most useful,

containing programs to handle name and address files, label printing and mail-merging.

Every program in this part of the book is accompanied by extensive documentation. And at least part of the function of these programs is to illustrate the use of various techniques in Wordwise Plus programming. In addition, Beverley provides a small number of short routines specifically intended to be incorporated into larger programs.



However, the fact that so much of the book consists of programs using the Wordwise Plus programming language I regard as something of a disappointment, especially in view of the book's title. I would have liked to see a number of chapters in this part of the book that were technique-orientated tutorials: perhaps a section on screen presentation and the use of VDU codes, one on string handling, one on machine code, one or two on the writing of utilities, and so on. Beverley does tell us about techniques in his very full treatment of each program presented, and there is clearly a very good case to be made for the lavish use of example programs in a book such as this. But a handbook needs to approach technique more directly for a greater part of its volume than the present work.

This is not to criticise the programs themselves. They cover a very wide range of topics, and there are many useful ones amongst them. Of course, Wordwise Plus is such an appealing language to write in that Beverley has fallen into the inevitable trap of writing some things in the language just for the sake of it, when it would have been much easier to use Basic. But even these programs serve the fundamental purpose of demonstrating the use of the language. So why not write an invoicing program in Wordwise Plus? - even if the triple-precision arithmetic necessary for calculating the VAT results in a much longer and slower program than could have been achieved in Basic: that is not the point!

To be quite fair, not every chapter in part two of the book presents program listings. The programs are sandwiched between two expository chapters: an introduction to the language, and a section on hints and tips for the Wordwise programmer. This latter contains one or two true gems: witness section 16.20 for example which gives some of the major zero page locations used by Wordwise Plus. These can be enormously useful to the programmer. Thus Beverley reveals that &8F contains a status byte of which bit zero tells you whether Wordwise is in Menu or Edit mode. This is just what you need if you have a program which has to issue commands directly from the Wordwise menu (Wordwise will not execute a command when in edit mode). This question is treated more fully in section 16.15.

The disc of the book is also available at a relativley modest £7.50. It contains some sixty programs which may be selected from a special menu - run of course from Wordwise Plus! A novel feature of the disc is that it offers a second option, which will unpack the files and store them on two new disc surfaces in a one-program-per-file format for easier subsequent retrieval.

All in all, Paul Beverley's glossy green tome will prove to be an extremely useful one for experienced users of Wordwise Plus. Less experienced users will benefit greatly from reading Bruce Smith's book before graduating to Beverley's Gargantuan Guide.

SPECIAL OFFER TO BEEBUG MEMBERS

Paul Beverly's book and accompanying disc are available at the special combined price of £19 plus £1.50 pp.

Orders must be received by 15th September 1986, and accompanied by membership number. Orders should be sent to:
Beebug Retail, Dolphin Place,
Holywell Hill,St Albans,
Herts AL1 1EX.

SOFTWARE FOR SIDEWAYS RAM (Part 3)

The final article from Bernard Hill on using sideways RAM software provides some useful utilities, including the ability to rename any sideways ROM command and avoid all those command-name conflicts.

In the first article of this series (Vol.5 No.2) we looked at the way in which a sideways ROM responds to service calls (in particular the *HELP service call - number 9) and produced some useful initialisation routines based on service call 1. Last month's article contained a complete * command parsing routine and illustrated this with * commands to drive a printer in its special effect modes. This month we expand the collection of * commands with some utilities.

These can be incorporated into your growing sideways RAM utility by merging the listing here with those of the previous two months. As with the code given last month, the ROM sections are completely independent, so you can omit or include whatever you wish. The only restriction is that you must place the calls to the two procedures somewhere between PROCromhead and PROCendrom.

The first routine, PROCutils, gives two fairly standard utilities: a function key lister and a ROM lister. The commands to call these utilities may be changed by altering the parameters of PROCutils. The function key lister is very standard, but the ROM lister has a feature not often seen elsewhere: the ability to distinguish between 8K and 16K ROMs.

The second routine, PROCredirect, needs more introduction. ROM clashes are a sad fact of life on the Beeb. Despite the BEEBUG convention of an optional prefix to distinguish the commands from ROMs of different companies, I always seem to have ROMs in my machine whose commands are 'hidden' by other ROMs. For example *EDIT is a command to which at least three ROMs respond - Disc Doctor, Toolkit Plus and Watford DFS.

This very easily customised routine enables you to rename those annoying commands. Now you can call View with *VIEW, or address your ROMs directly with a single command without worrying about their priorities at all.

As with the procedures in the previous articles, the one parameter is the line number of the first DATA statement of the redirection data. Each data item consists of an old command (e.g. the EDIT of *EDIT), the new command you want to use (e.g. *SHOW) and the name of the ROM which you want to respond to this (e.g. DISC DOCTOR). As long as you get the beginning of the ROM name correct and unambiguous then that will do, but be sure you have the correct case and spacing (e.g. for 'DISC DOCTOR', 'DI' would do, but 'Disc' would not). Consult the output of the ROMlist utility above if you don't know the exact name of any of your ROMs. Should you shuffle the ROMs in your machine then these routines will still work, but if you remove a ROM or mis-spell its name in the DATA statement then an error message is issued when you attempt to use it.

The command works by jumping directly into the relevant ROM, using JSR &8003. However, since you cannot jump into a ROM directly from another ROM, it is necessary to jump first into an area of normal memory. This means we must find a 'safe' area of normal memory, into which our ROM will put 20 bytes of code. The replacement command line also needs to be set up, which involves another free area to receive the reconfigured command (e.g. '*EDIT 10000').

Many areas are possible but be careful that any of the ROMs responding don't use this area for the command you want to process or the system may hang after the command has been executed. In the listing I've chosen the Tape Filing System BGET/BPUT workspace (&380-&3DF) but if you are likely to need to use this while the command is being executed (e.g. Toolkit Plus's *CHECK or MERGE) then you will need to choose some other area. Consider part of pages 6 or 7 (Basic use only), or 9, &A or &D. Simply change the variables 'target' and 'command' in line 20010.

ADDING TO THE ROUTINES

You will have noticed that the program built up in these three articles is highly

modular so that you can mix-and-match sections to suit yourself. Writers of other routines may care to conform to this format in order to share their work and ideas with other members through these pages.

Each section of code is contained in a single procedure which generates code to perform a specific action. Entry to the code is from the top and exit from the bottom (i.e. data bytes must be embedded and jumped over) so that the next procedure's code can start executing immediately. The entry will be with A, X, Y and the stack set up exactly as on entry to the ROM. Exit should be just the same, with any scratch space or memory areas used (e.g. &F2-3) completely restored and the stack intact so that the next section of code generated can behave as though it were the first on entry to the ROM.

The one exception to this may be if the ROM needs to signal back to the system that it wants to block all other ROMs (and routines) from having their go. In this case restore the stack, reload X and Y, then put Ø in A and RTS. Remember that 99% of the service calls passing through your ROM will not be for you and ignoring these simple rules can easily lead to a hung machine.

If you wish to use the command parser introduced last month (PROCcommandhandler) then organise your code along the models of this month and last month's utilities, with the parameter to PROCcommandhandler being the address of the command table (as explained last month). If you exit your code without executing a command then the procedure will handle this for you (it exits at the bottom), but if you do execute a command then be sure to put:

PLA:TAY:PLA:TAX:PLA:STA &A8:LDA#Ø:RTS

to finish securely.

We welcome the submission of further short routines for sideways ROM and RAM compatible with the format developed in these articles.

```
  156 CLEAR
  157 PROCutils("KEYS","ROMS")
  158 CLEAR
  159 PROCredirect(13000,"REDIRECT")
13000 REM redirection data
13010 DATA SHOW,EDIT,DISC DOCTOR
13020 DATA CHANGE,EDIT,TOOLKIT PLUS
13030 DATA SEEDISC,EDIT,DFS
13040 DATA VIEW,WORD,VIEW
13050 DATA "":REM end record
13060 :
13070 DEFPROCutils(key$,rom$)
13080 Q%=P%:R%=O%
13090 FOR opt=4 TO 7 STEP 3
13100 P%=Q%:O%=R%
13110 [ OPT opt
13120 JMP cmdhandler
13130 .utilcmds
13140 EQUS key$
13150 EQUW (keylist-1)MOD256*256+(keylis
t-1) DIV 256
13160 EQUS rom$
13170 EQUW (romlist-1)MOD256*256+(romlis
t-1) DIV 256
13180 EQUB Ø
13190 .keylist
13200 LDY #Ø
13210 .process
13220 LDX #Ø:LDA #255:STA &A8
13250 .findsmall
13260 LDA &BØØ,X:CMP &BØØ,Y
13270 BEQ ngt:BCC ngt
13280 CMP &A8:BCS ngt
13290 STA &A8
13300 .ngt
13310 INX:CPX #16:BNE findsmall
13320 LDA &A8:CMP #&FF:BNE skip
13330 LDA &BØØ,Y:STA &A8
13340 .skip
13350 JSR key
13360 LDA &BØØ,Y:TAX
13370 .nbyt
13380 CPX &A8:BEQ finikey
13390 LDA &BØ1,X:JSR write
13400 INX:JMP nbyt
13410 .finikey
13420 JSR &FFE7
13430 INY:CPY #16:BNE process
13440 JMP finiutil
13460 .write
13470 CMP #32:BCS notsmall
13480 PHA
13490 LDA #ASC"|":JSR &FFE3
13500 PLA:ADC #64:JSR &FFE3
13510 RTS
13520 .notsmall
13530 CMP #127:BNE notdelete
13540 LDA #ASC"|":JSR &FFE3
13550 LDA #ASC"?":JSR &FFE3
13560 RTS
13570 .notdelete
13580 BCS over128
13590 JSR &FFE3
13600 RTS
13610 .over128
```

```
13620 PHA:LDA #ASC"|":JSR &FFE3
13630 LDA #ASC"!":JSR &FFE3
13640 PLA:AND #&7F:JMP write
13650 .key
13660 LDX #0
13670 .next
13680 LDA ktitle,X:BEQ finitit
13690 JSR &FFE3
13700 INX:JMP next
13710 .finitit
13720 CPY #10:BCC lt9
13730 LDA #ASC"1":JSR &FFE3
13740 LDA #38:STA &A9
13750 JMP restout
13760 .lt9
13770 LDA #32:JSR &FFE3
13780 LDA #48:STA &A9
13790 .restout
13800 TYA:CLC:ADC &A9:JSR &FFE3
13810 RTS
13830 .ktitle
13840 EQUS " *KEY":EQUB 0
13870 .hex EQUS "0123456789ABCDEF"
13890 .romlist
13900 JSR printtitle
13910 LDY #&F
13920 .nextrom
13930 LDA #32:JSR &FFE3
13940 LDA hex,Y:JSR &FFE3
13950 JSR spaces
13960 TYA:PHA   save rom number
13970 LDA #6:STA &F6
13980 LDA #&80:STA &F7
13990 JSR &FFB9
14000 AND #&DF:CMP #&82:BEQ servrom
14010 CMP #&C2:BEQ langrom
14020 CMP #&40   Basic?
14030 BEQ langrom
14040   no rom
14050 PLA:TAY:JSR &FFE7
14060 JMP nexrom
14070 .servrom
14080 LDA #ASC"S":JMP outputrom
14090 .langrom
14100 LDA #ASC"L"
14110 .outputrom
14120 JSR &FFE3 : JSR spaces
14130 LDA #0:STA &F6
14140 LDA #&80:STA &F7
14150 .romcmp
14160 PLA:TAY:PHA   rom no
14170 JSR &FFB9:STA &A8
14180 LDA #&A0:STA &F7
14190 PLA:TAY:PHA
14200 JSR &FFB9:CMP &A8
14210 BNE rom16k
14220 INC &F6:BNE romcmp
14230 LDA #32:JSR &FFE3
14240 LDA #ASC"8":JSR &FFE3
14250 JMP kout
14260 .rom16k
14270 LDA #ASC"1":JSR &FFE3
14280 LDA #ASC"6":JSR &FFE3
14290 .kout
14300 LDA #ASC"K":JSR &FFE3
14310 JSR spaces
14320 PLA:TAY
14330 LDX #0
14340 .nextbytex
14350 TYA:PHA:TXA:PHA
14360 ADC #9:STA &F6
14370 LDA #&80:STA &F7
14380 JSR &FFB9:CMP #0:BEQ finrttl
14390 CMP #&D:BEQ finrttl
14400 JSR &FFE3
14410 PLA:TAX:PLA:TAY
14420 INX
14430 JMP nextbytex
14440 .finrttl
14450 JSR &FFE7
14460 PLA:TAX:PLA:TAY
14470 .nexrom
14480 DEY:CPY #&FF:BEQ dunromlist
14490 BEQ dunromlist
14500 JMP nextrom
14510 .finiutil
14520 .dunromlist
14530 PLA:TAY:PLA:TAX:PLA:STA &A8
14560 LDA #0:RTS
14590 .spaces LDA #32:JSR &FFE3:JSR &FFE
3:JSR &FFE3:RTS
14600 .rtitle EQUS "No. Type Size Title"
:EQUB 0
14610 .printtitle
14620 LDX #0
14630 .loopr
14640 LDA rtitle,X
14650 BEQ finrt
14660 JSR &FFE3
14670 INX
14680 JMP loopr
14690 .finrt
14700 JSR &FFE7:JSR &FFE7
14710 RTS
14720 .cmdhandler
14730 ]:NEXT opt
14750 PROCcommandhandler(utilcmds)
14760 PROCcommandhelp("UTILS",utilcmds)
14770 ENDPROC
14780 :
20000 DEFPROCredirect(dataloc,help$)
20010 target=&380:command=&394
20020 Q%=P%:R%=O%
20030 FOR opt=4 TO 7 STEP 3
20040 P%=Q%:O%=R%
20050 [ OPT opt
20060 JMP cmdhandler
20070 .newcmds
20080 EQUS STRING$(128,CHR$0)
20090 EQUS STRING$(128,CHR$0)
```

```
20100 .oldptrs
20110 EQUS STRING$(128,CHR$0)
20120 .oldcmds
20130 EQUS STRING$(128,CHR$0)
20140 EQUS STRING$(128,CHR$0)
20150 .romptrs
20160 EQUS STRING$(128,CHR$0)
20170 .romnames
20180 EQUS STRING$(128,CHR$0)
20190 EQUS STRING$(128,CHR$0)
20200 .redirect
20210 TYA:ASL A:TAY
20220 LDA oldptrs,Y:STA &A9
20230 LDA oldptrs+1,Y:STA &AA
20240 LDA romptrs,Y:STA &AB
20250 LDA romptrs+1,Y:STA &AC
20260 LDX #&FF:LDY #0
20270 .loop1 INX
20280 LDA (&A9),Y:BEQ endcopy
20290 STA command,X
20300 INY
20310 JMP loop1
20320 .endcopy
20330 TXA:PHA:TSX:LDY &102,X
20340 .loop2
20350 LDA (&F2),Y
20360 CMP #32:BEQ end2
20370 CMP #13:BEQ end2
20380 INY:JMP loop2
20390 .end2
20400 PLA:TAX
20410 .loop3
20420 LDA (&F2),Y
20430 STA command,X
20440 CMP #13
20450 BEQ end3
20460 INX:INY:JMP loop3
20470 .end3
20480 LDA #command MOD 256:STA &F2
20490 LDA #command DIV 256:STA &F3
20500 LDA #&80:STA &F7
20510 LDA #&10:STA &AE
20520 .nextrom
20530 LDA #9:STA &F6
20540 LDA #0:STA &AD
20550 DEC &AE
20560 BMI notfound
20570 .nextch
20580 LDY &AD
20590 LDA (&AB),Y
20600 BEQ found
20610 LDY &AE:JSR &FFB9
20620 LDY &AD:CMP (&AB),Y
20630 BNE nextrom
20640 INC &AD:INC &F6
20650 JMP nextch
20660 .notfound
20670 LDY #0
20680 .loop4
20690 LDA (&AB),Y:BEQ end4
20700 JSR &FFE3:INY
20710 JMP loop4
20720 .end4
20730 LDX #0
20740 .loop5
20750 LDA norom,X
20760 BEQ end5
20770 JSR &FFE3
20780 INX
20790 JMP loop5
20800 .end5
20810 PLA:TAY:PLA:TAX
20820 PLA:STA &A8:LDA #0:RTS
20830 .norom EQUS " not found"
20840 EQUW 13
20850 .found
20860 LDY #19
20870 .loop2 LDA code,Y:STA target,Y
20880 DEY:BPL loop2
20890 LDX &AE:LDY #0
20900 JSR target
20910 PLA:TAY:PLA:TAX:PLA:STA &A8
20920 LDA #0:RTS
20930 .code
20940 LDA &F4:PHA
20950 STX &F4:STX &FE30
20960 LDA #4:JSR &8003
20970 PLA:STA &F4:STA &FE30
20980 RTS
20990 .cmdhandler
21000 ]:NEXT opt
21010 PROCcommandhandler(newcmds)
21020 PROCcommandhelp(help$,newcmds)
21030 RESTORE dataloc
21040 n=0
21050 old=oldcmds-P%+O%
21060 rom=romnames-P%+O%
21070 new=newcmds-P%+O%
21080 oldptrs=oldptrs-P%+O%
21090 romptrs=romptrs-P%+O%
21100 REPEAT
21110 READ new$
21120 IF new$="" THEN 21270
21130 $new=new$:new=new+LENnew$
21140 ?new=(redirect-1) DIV 256
21150 new?1=(redirect-1) MOD 256
21160 new=new+2
21170 READ old$,rom$
21180 oldptrs?(2*n)=old MOD 256
21190 oldptrs?(2*n+1)=old DIV 256
21200 $old=old$:old=old+LENold$
21210 ?old=0:old=old+1
21220 romptrs?(2*n)=rom MOD 256
21230 romptrs?(2*n+1)=rom DIV 256
21240 $rom=rom$:rom=rom+LENrom$
21250 ?rom=0:rom=rom+1
21260 n=n+1
21270 UNTIL new$=""
21280 ENDPROC
```

SYSTEM DELTA

System Delta is much more than just a database. In fact it offers everything you need to build your own highly customised database system. How much of a landmark is this in the availability of serious database software for the Beeb? Geoff Bains reports.

Product : System Delta
Supplier : Minerva Systems
: 69, Sidwell Street,
: Exeter, Devon.
: Tel. 0392-37756
Price : £64.95 (ROM + Card Index)
: £19.95 (Reference manual)

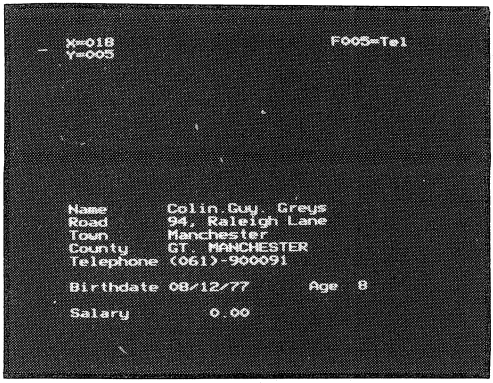
There are many database management programs around for the BBC micro. However, for an application requiring anything more than a simple card index, writing your own package is the only real answer. System Delta allows even the novice programmer to do just that.

System Delta comes on a 16K ROM - providing over 150 * commands - a full Card Index example program and a manual. The commands look after all the difficult aspects of writing your own database software - file handling, setting up, editing, and searching records.

A program written with System Delta commands can cope with databases of 8160 records, each of 8000 characters. Each record can contain up to 255 fields and each field 200 characters. The whole side of a disc is the only limitation in overall size of a file. Most importantly, System Delta can cope with five files at once so truly relational databases can be set up and interrogated.

System Delta is meant for a mode 7 display. Large records are viewed or edited by scrolling them in four directions. This is all automatic and requires very little programming from the user. The command *SDscroll, for example, will display a specified record and initiate the scrolling display.

To produce a simple browse facility on an existing file (called 'DATA') complete



with full scrolling across each card using the cursor keys, selection of the next or previous card using the 'N' and 'P' keys and 'F' to finish, this is all you need:

```
10 MODE7:*SDRESET
20 *SDOPEN DATA
30 *SDGREC 1
40 REPEAT:*SDSCROLL
50 IF G%=ASC"N" THEN *SDDIR 1
60 IF G%=ASC"P" THEN *SDDIR 0
70 IF G%<>ASC"F" THEN *SDNEXT
80 UNTIL G%=ASC"F":*SDCLOSE
```

A short program like this does not take long to write, nor does it take up much of the Beeb's valuable memory. However, the effect is truly professional.

The System Delta commands are divided into three groups. The first consists of named commands (all prefixed with 'SD'). These handle the file orientated aspects of programming. *SDopen, for example, will open a file. *SDedfield will edit a field.

The second group of commands are just numbered (again prefixed with 'SD'). These cover many useful features to help with creating a user interface to the file-handling operations. For example, *SD06 issues a 'Press a key' prompt, flushes the keyboard buffer and returns the ASCII code of the next key pressed in G%. *SD47 executes a * command.

The final group of commands are all *FX163,16 commands - ranging from *FX163,16,0 to *FX163,16,61. Acorn has specially allotted *FX163,16 to Minerva.

These commands are the really low level

commands that won't be required much by most users. They cover such features as converting an area of memory to upper case ASCII, and flushing the keyboard.

Most of the System Delta commands require parameters to specify the record, field, or whatever the command is to act on. Although the System Delta commands are all * commands they use Basic variables. Indeed, many System Delta commands return values into Basic resident (integer) variables for use by the programmer.

However, even this can be configured to your own preferences. If you are particularly fond of using, say, G% for some purpose of your own then the last key pressed can be redirected from G% to the variable key% simply by issuing the command *SDvar 3,key%.

The only aspect of System Delta that disappoints is the manual. The standard manual accompanying System Delta is mainly concerned with using the Card Index program. To create your own software (and this is really the whole idea) you'll need to fork out another £20 for the reference manual that explains all the commands. Even that leaves much to be desired. It is produced entirely from dot matrix printout which makes it very tiring to read.

This manual is extremely detailed - it goes right down to the level of the format of the data file - and there is not much you don't know about System Delta after wading through it. However, wade you must. It is clear that the manual was written by someone at Minerva very knowledgeable about the package with little thought for the struggling beginner. This is a shame as System Delta does enable a comparative programming beginner to produce excellent quality software, if only he can get to grips with Minerva's software.

All of which leads to the Card Index program supplied with the System Delta package. This is an application program written entirely in Basic using the System Delta commands. It does not exploit the full capabilities of System Delta - it only uses one file at a time - but it does show the kind of speed, quality and professionalism that can be achieved.

The Card Index is menu-driven with a series of interconnected menus, each controlling a different aspect of the database management - creating a file, editing, browsing, searching, etc.

The record card format is created using the ROM's built-in scrolling editor. Once the fields have been specified, their position and length is determined simply by moving around the card and typing in as many '@'s as are needed. Entering the data is also easy. Each card can be 'filled in' either from scratch, entering the contents of each field, or using a default card that is altered as required.

There is an excellent editing system for entering data. This allows full editing of a line as it is typed in. Fields on the card can even be assigned values dependent on other fields, even from other records. The age of each person in a personnel file could, for example, be displayed on each card with only the date of birth entered.

Once some records are entered into the file it can be browsed through or specific records displayed. The real power of the Card Index program comes when you want to search a file. Up to eight subsets of a file are created using equations to specify the rules of the selection. These equations can select records according to single or multiple criteria from the whole file, another subset, or from a combination of either. The access of the file is very fast. Minerva claims that any record in any length of file can be found in under three seconds.

The subset can then be printed out, either in full or just the fields that you require. The format of the printout can be different from that used for the record cards, indeed several printout formats can be used selectively.

The power of System Delta is clearly shown by this program. Indeed, Minerva is using System Delta to write its own specialized packages for sale. A full blown video rental control package has been written and an accounts package is under way. However, a glance through the Card Index listing reveals that it is not a package for the squeamish. Using System Delta is hard work but it is ultimately worth the trouble. System Delta has brought forward the art of database handling on the Beeb by a gigantic leap.

BEEBUG WORKSHOP BITMAPS

This month, Surac explains the use of bitmaps for speedy access to disc data.

A little while ago, I looked at various aspects of handling data files and saw that one limitation of the Beeb's memory is that any sensibly sized file must be held on disc. This, in turn, means that whenever a record is needed, it must be read from disc.

Now, reading data from disc is time-consuming. The process becomes particularly slow when we have to search files for records which satisfy specific conditions. Often, we have to look through the whole file to find what we want. There is a way around this problem, at least for a specific type of data. When an item of data can exist in just two states (e.g. Male/Female, Married/Single), we can hold a separate set of flags which code the status of that data item in every record.

Since each data item can only have 2 values, it is a "Boolean Variable" and can be coded as a single bit. For example, a bit set to "1" might mean that the corresponding record is for a man, while "0" means a woman. The coding is very compact - for instance, just 125 bytes can hold data about a given field in no fewer that 1000 records. Because bits give data about data, the technique is called "bit-mapping".

"So what?", you say. Because the bit maps hold data in such a compact form, they can be held in main memory. Hence, they can be searched far more quickly than disc-based files. We can identify, say, the record numbers of all married women, and then read only the data we need. There is no need to check and discard perhaps many hundreds of records to find just a handful.

A program using bit maps must reserve memory for the maps it needs:

```
100 MapSize%=(NoRecords% DIV 8)+1
110 DIM SMap% MapSize%,
        MMap% MapSize%
```

These two lines set up two byte-arrays, SMap% (Male/Female) and MMap% (Married/Single), each with at least NoRecords% bits. The "+1" in line 100 takes care of the case where NoRecords% is not an exact multiple of eight. In this example, I'm using two bit maps - you would, of course, create as many as you needed, with whatever names you wanted.

Having reserved memory for the maps, we must either set them up for a new or drastically altered system, or read them from disc into memory. To create them, check the relevant fields of every record, and set the bits accordingly as in PROCSetMaps.

Procedure SetMaps

```
5000 DEF PROCSetMaps
5010 LOCAL byte%,I%,mask%
5020 FOR I%=1 TO MapSize%:
       SMap%?I%=0:MMap%?I%=0:
       NEXT
5030 byte%=0
5040 mask%=1
5050 FOR I%=1 TO NoRecords%
5060   REM *** Get next record ***
5070   IF Sex$="M" THEN
          SMap%?byte%=SMap%?byte%
                              OR mask%
5080   IF Marr$="Y" THEN
          MMap%?byte%=MMap%?byte%
                              OR mask%
5090   mask%=mask%*2
5100   IF mask%>128 THEN mask%=1:
                         byte%=byte%+1
5110   NEXT
5120 ENDPROC
```

Zero the maps first (line 5020). The main work is then done by lines 5050-5110. Once the main data file has been opened for reading (OPENUP or OPENIN), the procedure reads each record in turn (line 5060). Then, depending on the sex and married state of the person in the record, it sets or doesn't set the associated bit in the 2 maps. The value of "mask%" cycles through 1..2..4..8...128..1.. to pick up each bit of a byte. The value of "byte%" goes up by one for each cycle of mask% to choose the relevant byte. Lines 5070 and 5080 control each map - you need similar lines for every map in use.

Creating the maps is slow, but you don't have to do it very often. Usually, you will set the bit(s) for one record at a time with a special procedure such as PROCSetBit. Note that this assumes that the records are numbered from 1 to NoRecords%.

```
            Procedure SetBit
10000 DEF PROCSetBit(Map%,Rec%,Flag%)
10010 LOCAL byte%,mask%
10020 mask%=2^((Rec%-1) MOD 8)
10030 byte%=(Rec%-1) DIV 8
10040 IF Flag% THEN
        Map%?byte%=(Map%?byte% OR  mask%)
      ELSE Map%?byte%=
             (Map%?byte% AND NOT mask%)
10050 ENDPROC
```

This procedure will set the bit corresponding to any record in any map to either 1 or 0 (TRUE or FALSE). To set our example maps for record number "n":

```
1000 PROCSetBit(SMap%,n,Sex$="M")
1010 PROCSetBit(MMap%,n,Marr$="Y")
```

In a similar way, we can search for specific records. First, a general purpose procedure to find and process all records satisfying a given condition:

```
6000 DEF PROCFindRecs(Cond$,Action$)
6010 LOCAL byte%,dy%,I%,mask%
6020 byte%=0
6030 mask%=1
6040 FOR I%=1 TO NoRecords%
6050    IF EVAL(Cond$) AND mask% THEN
           dy%=EVAL(Action$)
6060    mask%=mask%*2
6070    IF mask%>128 THEN mask%=1:
                          byte%=byte%+1
6080    NEXT
6090 ENDPROC
```

As in PROCSetBit, it goes through the bits representing every record. However, being general purpose, it does not define the search conditions or what to do when it finds suitable records. It expects these to be supplied as strings.

"Cond$" defines which maps to check, and how (e.g. AND or OR) they are to be combined. Although the PROC is clever enough to mask the bits in each byte, you must show it how to use byte% to select the appropriate byte in the map(s) as it cycles through them.

During the search, whenever the procedure identifies a suitable record, it EVALuates "Action$" and assigns its value to the dummy dy% (line 6050). This provides a way to pre-program different record-handlers for different searches, and to pass the appropriate one to PROCFindRecs. Define the handlers as FuNctions, with the exit point returning a dummy value (e.g. =0):

```
20000 DEF FNHandle1
20010 REM *** Read Record I% ***
20020 REM *** Process it ***
20030 REM *** Re-write to file ***
20040 =0:REM *** Dummy value for exit
```

It's really a procedure, but they can't be EVALuated.

Suppose you wished to search for all married women, and read and process each such record with FNHandle1. You would get into the procedure with:

```
1100 PROCFindRecs("NOT SMap%?byte% AND
     MMap%?byte%","FNHandle1")
```

Notice how the search is defined. The bits in SArr% are SET for men, so we need "NOT set" to find women; MArr% bits are set for married. The "?byte%" provides the procedure with a way of searching through the maps.

That's a way of searching all the records. To check a particular record:

```
11000 DEF FNReadBit(Map%,Rec%)
11010 LOCAL byte%,mask%
11020 mask%=2^((Rec%-1) MOD 8)
11030 byte%=(Rec%-1) DIV 8
11040 =((Map%?byte% AND mask%)>0)
```

→61

POSTBAG

POSTBAG

DISC DRIVES NOT WHAT THEY SEEM

In the past year I have purchased or had replaced four disc drives all under the same very well known brand name, each of which has turned out to be of a different make internally.

The first point is the common practice of advertising these drives purely under a supplier's trade name, giving the impression that they are made by the supplier, instead of in the Far East which is the case. This no doubt enables the supplier to switch the internal works to whatever is cheapest or most readily available.

Different drive designs have different specifications which need to be matched to the internal keyboard links 3 & 4. The fact is that some drives are faster, and therefore better than others.

There is a need for more honesty and help for purchasers of disc drives. Suppliers should state the name of the drive as well as their own name. There should also be be some coding, using the links 3 & 4, to enable one to know how fast the drive will be, and whether it will match one's existing drive when adding a second.

E.A.Allchin

Mr Allchin has highlighted some important points. Major suppliers in the UK purchase disc drives in quantity to be packaged up for the home market. These disc drives are sold in turn through many dealers and computer shops, including BEEBUG's own showroom. There appears at present to be a relative shortage of disc units arriving in the UK and major suppliers are unable to specify in advance what make of drive will be supplied. In turn, shops like our own have to accept what they are supplied with, if they are not to wait excessive periods of time for delivery.

If members want a specific type of drive we suggest they either phone several shops or dealers to see what is immediately available, or be prepared to wait, possibly some considerable time, for the unit of their choice.

Some drives certainly are faster than others, but only if the internal links are correctly set. We hope shortly to publish information on this subject that will act as a guide for BEEBUG members.

LIGHT ON THE SHADOWS

Can you please explain the meanings of, and differences between, sideways RAM, sideways ROM and shadow RAM?

G.D.Alliban

We often receive letters asking about the various add-on RAM and ROM boards particularly for the model B. The standard Beeb uses up to 20K of memory to store the screen display, often severely limiting the amount of memory available to software (with Wordwise or View for example), or for your own programs. Shadow RAM (built in on the B+ and Master), such as the Aries B-32 or Watford 32K RAM board, provides extra memory so that applications or user software have the maximum 24K (approximately) of memory always available, irrespective of the mode.

Next, the Beeb has done much to popularise the use of software in ROM or EPROM form. The Beeb has empty sockets for two additional 'sideways' ROMs (after OS, DFS, Basic) but is designed to handle up to a total of 16. A sideways ROM board effectively adds extra ROM sockets to your Beeb.

Thirdly, ROM software can also be saved on disc as a ROM image and loaded into sideways RAM. Thus ROM software is no longer limited by the number of physical sockets available. Many sideways ROM boards, like that from ATPL sold by BEEBUG, allow some of the ROM sockets to be filled with RAM, not just ROMs.

A further possibility, not mentioned by Mr Alliban, is the so-called silicon disc, a bank of memory chips that looks like a disc unit to the computer but very much faster in operation.

This is a somewhat simplified picture but does summarise the main choices available to those who wish to enhance their model B (closer to the capability of the B+ or Master which have shadow RAM and sideways RAM built in).

HINTS and tips

Determining Mode Legally

Although reading memory location &355 will reveal the display mode currently in use, the correct and legal way to do this (that will work on the Master too) is to use the OSBYTE 135 routine. For example:

```
100 A%=135
110 mode%=((USR &FFF4)
AND &FF0000) DIV &10000
```

Jagdish Sah

Wordwise Plus Line

To draw a horizontal line across your paper that conforms to the line length, margins, and so on, the following simple line should be entered. Note that (f1) is the green embedded command and (f2) the white one.

```
(f1)us(f1)fi(f1)ue(f2)
```

Richard Sterry

Slow Writing

This short routine intercepts the WRCHV (write character vector) to delay the printing of characters for the slow appearance of instructions for games and similar applications. The routine can be easily relocated to other areas of memory by altering line 10. The effect can be switched off and on with ?&70=0 and ?&70=1, respectively.

```
10 PROG%=&900
20 FOR pass=0TO3 STEP3
30 P%=PROG%
40 [OPT pass
50 PHA:TXA:PHA:TYA:PHA
60 LDA &70:CMP #0
                :BEQ EXIT
70 LDX #40
80 .CONT LDY #255
90 .LOOP DEY:CPY #0
                :BNE LOOP
100 DEX:CPX #0:BNE CONT
110 .EXIT
120 PLA:TAY:PLA:TAX:PLA
130 JMP ?&20E+256*?&20F
140 ]NEXT pass
150 ?&70=255
160 ?&20E=PROG% MOD 256
170 ?&20F=PROG% DIV 256
```

C.D. Reynolds

Star Command Parameters

To pass the values of Basic variables as parameters to * commands you should use the OSCLI command (not available in Basic I). For example, to catalogue each disc drive in turn:

```
100 FOR N%=0 TO 3
110 OSCLI("CAT "+STR$(N%))
120 IF GET
130 NEXT N%
```

Brian Clarkson

Aries Boot

A !BOOT file to turn off the Aries B-32 shadow RAM before loading and running a program cannot use the normal Aries command *XOFF to disable the screen RAM as this requires the user to press Break. Instead the command *XOFF 1 should be used. For example the !BOOT file could contain:

```
*XOFF 1
*KEY 10CHAIN"menu"|M
CALL !-4
```

Cursor Position in Wordwise Plus

There is an alternative to determining the horizontal cursor position from location &7E that is less likely to change with new versions of Wordwise Plus. The VDU workspace location, &318 will return the current horizontal position of the cursor. Similarly &319 contains the vertical position of the cursor which may be used to determine whether the word processor is in edit mode (?&318=13) or not.

Michael Dowling

Software Tube Turn-off

The Tube can be turned off without touching the mains switch with the following short program:

```
100 *K.0 *FX234|M*B.|M*
T.|MA%=144:X%=1:CALL&FFF4|M
*D.|M*TV|M
110 *FX138,0,128
120 END
```

Pressing break will restore the Tube with any program residing in the second processor intact.

Guttorm Vik

FALSE, TRUE and True

If you are using variables as logical flags, and merely testing along the lines of IF X% THEN then you must ensure that you always stick to values of -1 and 0 for these variables (TRUE and FALSE as defined by Basic). If you use other values (which will register as true and false) you can run into problems using NOT X% if X% is not -1. The reason for this is that only a value of -1 will go to 0 when the NOT operator is used, and thus the logical condition would not otherwise be reversed.

D. Morgan

1st course

Getting it Right

David Graham explores the use of the GET function, a most useful weapon when dealing with keyboard input.

In simple terms both GET and GET$ "get" a character from the keyboard - or more strictly speaking from the keyboard buffer: a distinction which we will deal with in a moment. GET$ returns an actual character in the form of a string of length one; while GET returns a number corresponding to the ASCII value of the key pressed. Thus for example, if the key pressed is "L", GET$ would return "L", and GET would give 76, the ASCII value of "L". Both the User Guide and our Giant Reference Card contain ASCII tables for reference.

You can test out these two functions very easily by just typing:

```
PRINT GET
```

or PRINT GET$

If you do this you will see the machine appear to hang after Return is pressed until you press any other key. You will then see either the ASCII value or the actual letter of the key pressed, displayed on the screen. In normal use, these two functions are read into a variable - a numeric variable in the case of GET, and a string variable with GET$. Thus a key could be read into the variable A with:

```
A=GET
```

or into A$ with:

```
A$=GET$
```

The surrounding program can then test A or A$ for various conditions and respond accordingly.

Perhaps the most obvious use for the GET and GET$ functions is to handle the input of just a single character from the keyboard, such as you might need in the case of a Yes / No question, or if a program was required to wait until the space bar was pressed for example. GET scores heavily over the INPUT statement in such cases, since it requires no pressing of Return to enter the answer, so that a response can be given with a single key-press.

In fact GET and GET$ are also often used for much more complex input when the programmer needs keyboard input to be handled in a different way from the fixed format of the INPUT statement. For example, you might require an input string to be numerical only; but with the INPUT statement, the user could enter a whole string of letters, and the program would not know about it until Return was pressed at the end of the string. But using GET or GET$ the programmer can check each key as it is pressed to see whether it corresponds to what his program requires. Of course, the programmer has to do a little more work with this approach, creating a string out of individual characters, handling incorrect input etc.

GETTING THE SPACE BAR

For the moment we will look at the two examples mentioned above, taking the simplest of these first. The object of the exercise is to wait for the space bar to be pressed, so that the user may read a screen of text, or whatever, and call up the next part of the program when he is ready. The routine is simple:

```
100 REM PROG 1: GET THE SPACE BAR
110 PRINT"Press Space Bar to Continue"
120 REPEAT: UNTIL GET=32
130 PRINT"Program now continues"
```

Line 120 just keeps looking at any key pressed until a key with ASCII value 32 (i.e. the space bar) is returned. The program then moves on. We could equally well have used GET$. Line 120 would then be:

```
120 REPEAT: UNTIL GET$=" "
```

Here the line checks to see if the character pressed matches the one in quotes: a space in this case. Alternatively, if you replace line 120 as follows:

```
120 A=GET
```

Then the program will wait until ANY key is pressed.

FLUSHING THE BUFFER

This routine lacks one refinement which is quite important, but which is too often overlooked in professional software: flushing the buffer. When you press any key on the BBC micro, a code for the key is entered into what is called the keyboard buffer. This is an area of memory which can accommodate a whole series of key codes, until the computer has time, or is directed, to deal with them.

The GET and GET$ functions, and the INPUT statement all get their key-pressed information from the keyboard buffer, and not directly from the keyboard itself. Because of this it is important to "flush" the keyboard buffer of all characters previously placed in it before using INPUT, GET or GET$. This is done with an FX call (*FX15,1). We can see this in action in the next example.

DETECTING YES / NO ANSWERS

One of the most commonly used computer responses is of the Yes / No type, and the GET or GET$ function can handle this with ease, as can be seen from the following example:

```
100 REM PROG 2: Yes / No Answer
110 REPEAT
120 PROCgame
130 PROCanswer
140 UNTIL reply$<>"Y"
150 PRINT'"Bye for now"
160 END
170 :
180 DEFPROCgame
190 PRINT'"***************"
200 PRINT "***THE GAME***"
210 PRINT "***************"
220 ENDPROC
230 :
240 DEFPROCanswer
250 PRINT
260 PRINT"Another game ?"
270 *FX15,1
280 reply$=GET$
290 ENDPROC
```

The main part of this program consists of a REPEAT loop in lines 110 - 140 which repeatedly calls the two procedures PROCgame (which would normally contain a real game), and PROCanswer. This latter uses GET$ to check whether the user wants to play the game again or not. The reply to the question is a single key-press, and this is put into the string variable reply$. The main program loop checks this, and exits if the user presses any key other than "Y".

RESPONDING TO MENUS

Another frequently used application of the GET function is in multiple choice menus. Here a single key-press is again used, but this time there may be quite a considerable number of "correct" responses, each requiring a different action. Again we can use either GET or GET$, and really there is little to choose between them. Program 3 gives an example of this kind of menu.

```
100 REM PROG 3: GENERAL PURPOSE MENU
110 REPEAT
120 MODE7
130 PROCdisplay
140 PROCchoice
150 UNTIL A=88
160 PRINT:END
170 :
180 DEFPROCdisplay
190 PRINTTAB(10,2)CHR$134;CHR$141;;"TE
ST MENU"
200 PRINTTAB(10,3)CHR$134;CHR$141;;"TE
ST MENU"
210 PRINTTAB(5,6)CHR$131;"A. First Cho
ice"
220 PRINTTAB(5,8)CHR$131;"B. Second Ch
oice"
230 PRINTTAB(5,10)CHR$131;"C. Third Ch
oice"
240 PRINTTAB(5,14)CHR$131;"*  Star Com
mand"
250 PRINTTAB(5,16)CHR$131;"X. Exit"
260 ENDPROC
270 :
280 DEFPROCchoice
290 REPEAT
300 PRINTTAB(0,20)"Please enter your c
hoice  ";
310 *FX15,1
320 A=GET AND &DF:IF A=10 THEN A=42
330 UNTIL A=42 OR A=88 OR A>64 AND A<6
8
340 IF A=42 THEN PROCstarcommand
350 IF A=65 THEN PROCfirst
360 IF A=66 THEN PROCsecond
370 IF A=67 THEN PROCthird
380 ENDPROC
390 :
400 DEFPROCstarcommand
410 INPUT'"*"A$:PRINT
420 $&A00=A$:X%=0:Y%=&A:X=USR(&FFF7)
```

```
430 PROCspace
440 ENDPROC
450 :
460 DEFPROCspace
470 PRINT'"Press spacebar to continue
";
480 REPEAT:UNTIL GET=32
490 ENDPROC
500 :
510 DEFPROCfirst
520 CLS:PRINT'"First choice"
530 PROCspace
540 ENDPROC
550 :
560 DEFPROCsecond
570 CLS:PRINT'"Second choice"
580 PROCspace
590 ENDPROC
600 :
610 DEFPROCthird
620 CLS:PRINT'"Third choice"
630 PROCspace
640 ENDPROC
```

The program is organised around a main REPEAT loop on lines 110 - 150. This in turn calls just two procedures. PROCdisplay prints the menu options on the screen, and PROCchoice handles the keyboard input. In the current example there are three main choices, each calling a dummy procedure. In addition, the menu has an exit option and will also handle star commands.

PROCchoice is constructed around the GET function on line 320. This uses a programming trick to ensure that it responds to key-presses irrespective of case. If you doubt the wisdom of this, try entering:

```
PRINT CHR$(GET AND &DF)
```

directly from the keyboard. It will return upper case letters regardless of whether you enter them in upper or lower case. Line 320 uses the same principle.

The test for the key-press in PROCchoice is performed within a REPEAT loop (lines 290 - 330). An exit occurs only when one of the designated keys (A, B, C, * or X) is pressed. Lines 340 - 370 then check which of these five selections was made, and react accordingly. If * is chosen (ASCII 42), PROCstarcommand is called. This short procedure allows any star command (e.g. *CAT etc) to be issued from the menu, and because it avoids the use of OSCLI, it will work equally well on BASIC I and II. Once the star command has been performed, a press of the space bar (PROCspace) takes you back to the menu. You should bear in mind when executing star commands from the menu that some, like *COPY, will destroy the menu program, since they overwrite user memory.

Selecting A, B, or C from the menu causes the program to enter one of three dummy procedures at lines 510, 560 or 610. These would normally form the essence of the whole program. You may have noticed that the main testing loop which checks to see which key was pressed, appears to ignore the "X" option (A=88). In fact if A=88, this value is returned to the main loop where it causes an exit on line 150.

MODIFYING THE MENU

The menu presented here should be fairly easy to tailor to your own requirements. It would be a simple matter to include many more different items for selection, by increasing the value 68 in line 330, and by including tests for the new values of A just after line 370. A further useful enhancement would be the addition of a proper error handling routine. But this is beyond the scope of the present article.

In the next issue, First Course will be exploring the related input function, INKEY, in all its various guises.

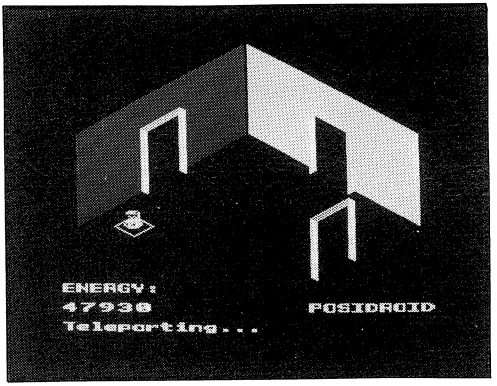
POINTS ARISING POINTS ARISING POINTS ARISING POINTS

BEEBUG FILER GRAPH OPTION (BEEBUG Vol.5 No.2)

The magazine/cassette disc for Vol.5 No.2 contains an error in the Filer Graphics program that prevents compatibility with the original Filer program. Change the value assigned to FDR in line 1040 from 512 to 256. Later copies of this magazine cassette/disc (from July onwards) contain a corrected version. The program is listed correctly in the magazine.

POSIDROID

Can you guide the Posidroid robot successfully throught the three-dimensional labyrinth, exterminate the mutants, and save the world? Jonathan Temple reveals his latest game, a thought-provoking 3D graphical adventure.



Long, long ago, in a distant century, the people of Earth built a huge nuclear power station. As the years passed, the station functioned reasonably successfully, until a new, better fuel source was discovered and the station was closed down.

This new fuel was used for many years, but eventually supplies began to run out and an alternative had to be sought. It was decided to re-open the nuclear station, but there was a problem...scientists scanning the station discovered that radiation had leaked and had spread throughout one of the main buildings. Mutant creatures had developed, and the area was unsafe for any human being to venture into.

Some way of exterminating the creatures was needed - the radiation could be cleared later by the scientists. Temporarily at a loss, the scientists turned in desperation to old records of the station made at the time of its closure. They discovered that there was a remote-control 'Posi' droid (especially designed for working in high levels of radiation) lying dormant in one of the rooms. The droid was re-activated and a controller chosen to guide it through the building. After much deliberation it was announced that YOU were going to be assigned this task...

The droid is controlled using the 'A','Z','*' and '?' keys on your console. Objects seen in any room can be picked up by simply guiding the droid over them, but the droid can only hold one object at a time. Any object currently held will be swapped for the one picked up, the currently-held object being displayed at the bottom of your monitor. The 'Return' key should be used to fire - that is, only if you do well enough to find the object necessary!

Unfortunately the droid is an old one, and over the years, its energy reserves have gradually diminished. Energy is depleted by moving, colliding with the monsters inhabiting the station or by teleporting using the several teleport pads which are spread throughout the building; if the droid runs out of energy the mission must be aborted. However, to help you, the droid's energy can be at least partly recharged by pressing the 'R' key - again, only if you're carrying the right equipment!

In addition you can find out how well you're doing by pressing the 'S' key, which will show you in percentage form how much of the game you have completed. Another useful key is 'P' which will pause the game; whilst the game is paused 'S' and 'Q' can be used to turn the sound on and off respectively. 'C' will allow the game to be continued.

This type of game is always more fun if you explore and discover things for yourself, so I have tried to give you very few hints. However, remember that for the game to be successfully completed making a map is almost essential!

The program should be fairly straightforward to enter, but remember the cardinal rule of typing in games and MAKE SURE that you save the program before

attempting to run and debug it. This must be done because of the indirection operations used in the program, as there is a chance that you may find that the program is corrupted if you make any typing errors.

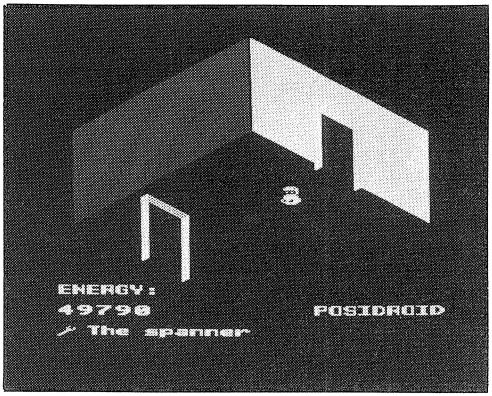
Also, when entering the program do not type in extra spaces or the ones after the line numbers, or you may find yourself suffering from the dreaded 'No room' error. If this does happen during the course of playing the game, then try missing out unnecessary lines such as those containing only colons, or the REM statements in lines 10-60.

Model B disc users, or other people using a machine with PAGE set any higher than &E00 (without the use of shadow RAM), will need to include a suitable downloading routine as given below:-

```
1 IF PA.<&E01 THEN 10
2 *K.0 *T.|MF.A%=0TO(TOP-PA.)S.4
  :A%!&E00=A%!PA.:N.|MPA.=&E00|M
  O.|MDEL.1,4|MRUN|M
3 *FX 138,0,128
4 END
```

If you are using this routine be especially careful to save the program first, as again your program could be corrupted, for instance perhaps by pressing Break.

```
 10 REM PROGRAM POSIDROID
 20 REM VERSION B1.8
 30 REM AUTHOR  J.Temple
 40 REM BEEBUG  AUG/SEPT 1986
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
100 MODE 2:ON ERROR GOTO 3890
110 PROCinit:PROCchars
120 PROCenvs:PROCtitle
130 REPEAT:MODE 2
140 PROCsetvars:PROCscreen
150 REPEAT:PROCdroid
160 IF F% PROCfiring
170 IF A%>-1 PROCmonster
180 UNTIL E%<5 OR C%=16
190 IFC%=16 PROCcongrats ELSE PROCend
200 UNTIL FALSE
210 :
1000 DEFPROCdroid
1010 V%=X%:W%=Y%:U%=Z%
1020 IFINKEY-56 PROCpause
1030 IFINKEY-52 IFH% IFH%<5 PROCinf("Re
charging..."):FOR N%=1 TO150:SOUND 3,3,2
00,1:PROCenergy(-100):NEXT:N%(H%,1)=64:H
%=0:PROCheld
1040 IFINKEY-74 IFH%=11 IFF%=0 J%=X%:K%
=Y%:SOUND 17,1,200,3:Q%=(Z%=237)-(Z%=226
):R%=(Z%=239)-(Z%=224):F%=1:PROCplot(J%,
K%,228)
1050 IFINKEY-82 PROCinf("You have compl
eted#"+STR$(FNcalc)+"% of the game"):PRO
Cheld
1060 IFINKEY-66 Z%=237:IFU%=237 X%=X%-1
1070 IFINKEY-98 Z%=226:IFU%=226 X%=X%+1
1080 IFINKEY-73 Z%=239:IFU%=239 Y%=Y%-1
1090 IFINKEY-105 Z%=224:IFU%=224 Y%=Y%+
1
1100 IF X%<0 OR X%>8 OR Y%<0 OR Y%>8 PR
OCedges
1110 M%=0:IF X%<>V% OR Y%<>W% M%=1
```




```
1120 IF M% OR Z%<>U% PROCdr(V%,W%,U%):P
ROCdr(X%,Y%,Z%):IF M% PROCenergy(5):IF X
%=1 IF Y%=7 IF T%(L%,0)=P% PROCteleport
1130 IFS% IFM% IFX%=4 IFY%=4 PROCcheck
1140 ENDPROC
1150 :
1160 DEFPROCpause
1170 PROCinf("Paused")
1180 REPEAT A=GET AND &DF
1190 IFA=81 PROCinf("Sound off"):*FX210
,1
1200 IFA=83 PROCinf("Sound on"):*FX210
1210 UNTIL A=67 OR A=70
1220 IF A=70 E%=0 ELSE PROCheld
1230 ENDPROC
1240 :
1250 DEFPROCfiring
1260 PROCplot(J%,K%,228)
1270 IFJ%=A% IFK%=B% GOTO1300
1280 J%=J%+Q%:K%=K%+R%
1290 IFJ%<0 OR K%<0 OR J%>8 OR K%>8 F%=
0:ENDPROC
```

```
1300 IFJ%=A% IFK%=B% PROCplot(A%,B%,236
):SOUND 16,-12,6,2:A%=-1:C%=C%+1:F%=0:G%
?P%=0:ENDPROC
1310 PROCplot(J%,K%,228)
1320 ENDPROC
1330 :
1340 DEFPROCedges
1350 X%=V%:Y%=W%
1360 IFX%=0 IFY%=4 IF(d% AND2) PROCnew(
-1):ENDPROC
1370 IFX%=8 IFY%=4 IF(d% AND1) PROCnew(
1):ENDPROC
1380 IFY%=0 IFX%=4 IF(d% AND8) PROCnew(
-4):ENDPROC
1390 IFY%=8 IFX%=4 IF(d% AND4) PROCnew(
4):ENDPROC
1400 Z%=U%
1410 ENDPROC
1420 :
1430 DEFPROCnew(N%)
1440 IF N%=1 IF T% PROCdoor:ENDPROC
1450 IFABS(N%)<4 X%=8-X% ELSE Y%=8-Y%
1460 P%=P%+N%:PROCscreen
1470 ENDPROC
1480 :
1490 DEFPROCdoor
1500 IF (H%-4=T%) OR (T%=6 AND H%=8) D%
(L%,1)=0:P%=P%+1:X%=0:PROCscreen:PROCinf
("Key "+STR$(H%-4)+" opened door") ELSE
PROCinf("This door is locked")
1510 PROCheld
1520 ENDPROC
1530 :
1540 DEFPROCcheck
1550 IFS%=12 PROCcomp:ENDPROC
1560 IFS%=13 PROCcons:ENDPROC
1570 SOUND 18,2,100,3
1580 PROCobj(S%):N%(S%,1)=64:p%=0
1590 IFH% PROCobj(H%):N%(H%,1)=P%:A=H%
1600 H%=S%:S%=A:PROCheld
1610 ENDPROC
1620 :
1630 DEFPROCcomp
1640 IFH%<>9 PROCinf("The security comp
uter#cannot be moved"):PROCheld:ENDPROC
1650 PROCinf("You insert the disc and#w
atch...")
1660 r%?10=55:N%(9,1)=65:H%=0:PROCscree
n
1670 PROCinf("...as a secret door is#re
vealed")
1680 PROCheld
1690 ENDPROC
1700 :
1710 DEFPROCcons
1720 IFH%<>10 OR t% PROCinf("The telepo
rt console#cannot be moved"):PROCheld:EN
DPROC
1730 PROCinf("You use the spanner to#re
pair the broken console")
1740 t%=TRUE:PROCheld
1750 ENDPROC
1760 :
1770 DEFPROCteleport
1780 IF t%=0 PROCinf("The teleport does
not#appear to be functioning"):PROCheld
:ENDPROC
1790 VDU19,3,14;0;
1800 PROCinf("Teleporting...")
1810 FOR A%=1 TO 100:PROCenergy(50)
1820 IFA%=40 PROCdr(X%,Y%,Z%)
1830 SOUND 3,3,1,1:NEXT:VDU19,3,3;0;
1840 P%=T%(L%,1):PROCscreen
1850 ENDPROC
1860 :
1870 DEFPROCmonster
1880 IFA%=X% IFB%=Y% PROCenergy(1000):S
OUND 3,3,25,2
1890 D%=D%+1:IF D%<3 ENDPROC
1900 D%=0:PROCplot(A%,B%,236)
1910 IFA%=X% IFB%<Y% B%=B%+1 ELSE IFB%>
Y% B%=B%-1
1920 IFA%<X% A%=A%+1 ELSE IF A%>X% A%=A
%-1
1930 PROCplot(A%,B%,236):SOUND 3,3,1,2
1940 IFA%=X% IFB%=Y% PROCenergy(1000):S
OUND 3,3,25,2
1950 ENDPROC
1960 :
1970 DEFPROCinf(T$)
1980 VDU 4,17,7,31,0,29
1990 PRINTSPC(24);TAB(0,29);
2000 N%=INSTR(T$,"#")
2010 IFN% PRINT LEFT$(T$,N%-1);TAB(0,31
);MID$(T$,N%+1); ELSE PRINT T$
2020 IFLEN(T$)>15 THEN TIME=0:REPEAT UN
TIL TIME>LEN(T$)*10
2030 VDU 5
2040 ENDPROC
2050 :
2060 DEFPROCdr(X%,Y%,Z%)
2070 N%=X%+Y%*9
2080 MOVE 1120-a%?N%*64,716-b%?N%*32
2090 VDU Z%,10,8,Z%+1
2100 ENDPROC
2110 :
2120 DEFPROCobj(N%)
2130 GCOL3,N%(N%,2) EOR4
2140 PROCplot(4,4,N%(N%,0)+229)
2150 GCOL3,8
2160 ENDPROC
2170 :
2180 DEFPROCplot(A%,B%,C%)
2190 N%=A%+B%*9
2200 MOVE 1120-a%?N%*64,700-b%?N%*32
2210 VDU C%
2220 ENDPROC
2230 :
2240 DEFPROCheld
2250 VDU 4,17,N%(H%,2),31,0,29
```

```
 2260 PRINT SPC(77);TAB(0,29);
 2270 IFH% VDU N%(H%,0)+229,9,17,7:PRINT
$(n%+N%(H%,0)*32);:IF N%(H%,0)=1 VDU9,44
+H%,32
 2280 VDU 5
 2290 ENDPROC
 2300 :
 2310 DEFPROCenergy(N%)
 2320 E%=E%-N%:IFE%<0 E%=0
 2330 VDU 4,17,7,31,0,27:?&34F=32
 2340 PRINT LEFT$("00000",5-LEN(STR$(E%)
))+STR$(E%);" ";
 2350 ?&34F=24:VDU 5
 2360 ENDPROC
 2370 :
 2380 DEFFNcalc
 2390 A=0:FOR N%=0 TO 3
 2400 IF D%(N%,1)=0 A=A+8
 2410 NEXT
 2420 =A+C%*3-t%*4-(N%(9,1)=65)*16
 2430 :
 2440 DEFPROCend
 2450 ?&34F=32
 2460 VDU4,28,4,17,16,11,12,26,5
 2470 PROCprint("GAME OVER",384,636,1,3)
 2480 PROCprint(STR$(FNcalc)+"% complete
",288,572,4,6)
 2490 PROCprint("<SPACE>",448,508,4,5)
 2500 *FX15,0
 2510 REPEAT UNTIL GET=32
 2520 ENDPROC
 2530 :
 2540 DEFPROCcongrats
 2550 TIME=0:REPEAT UNTIL TIME>200
 2560 GCOL 0,0:VDU 29,640;512;:MOVE 0,0
 2570 FOR A=0 TO 6.3 STEP .1:MOVE 0,0
 2580 PLOT 85,896*SIN(A),896*COS(A)
 2590 NEXT:VDU29,0;0;:*FX21
 2600 PROCprint("CONGRATULATIONS!",128,8
92,1,3)
 2610 PROCprint("You have completed",64,
764,4,6)
 2620 PROCprint("a difficult mission",32
,700,4,6)
 2630 PROCprint("<SPACE>",416,604,4,5)
 2640 REPEAT UNTIL GET=32
 2650 ENDPROC
 2660 :
 2670 DEFPROCprint(T$,X,Y,A,B)
 2680 GCOL0,A:MOVE X,Y:PRINT T$
 2690 GCOL0,B:MOVE X-8,Y-4:PRINT T$
 2700 ENDPROC
 2710 :
 2720 DEFPROCsetvars
 2730 VDU 23;10,32;0;0;0;:RESTORE3870
 2740 FOR N%=1 TO 13
 2750 READ N%(N%,0),N%(N%,1),N%(N%,2)
 2760 NEXT:FOR N%=0 TO 3
 2770 READ D%(N%,0),D%(N%,1):NEXT
 2780 FOR N%=0 TO 15:!(G%+N%*4)=0:NEXT
 2790 FOR N%=1 TO 16
 2800 READ R%:G%?R%=1:NEXT
 2810 FOR N%=8 TO 15
 2820 READ A%:VDU 19,N%,A%;0;:NEXT
 2830 ?&34F=24:?&30A=25
 2840 P%=21:H%=0:E%=50000:C%=0
 2850 X%=4:Y%=4:Z%=226:t%=0
 2860 $r%="537284965F3E9A8816525FB6C93E9
32845729EC448DA93A01336053A5B7281A0"
 2870 ENDPROC
 2880 :
 2890 DEFPROCscreen
 2900 T%=0:d%=EVAL("&"+CHR$(r%?P%))
 2910 L%=P% DIV16:VDU 4,17,15,12,31,0,25
 2920 PRINT"ENERGY:";TAB(13,27)"POSIDROI
D";
 2930 PROCenergy(0):PROCheld
 2940 RESTORE 3820
 2950 FOR N%=1 TO 18:READ D%,A%
 2960 IF D% READ B%:PLOT D%,A%*64,28+B%*
32 ELSE GCOL 0,A%
 2970 NEXT
 2980 IF T%(L%,0)=P% PROCpad(3)
 2990 IF R%(L%)=P% PROCpad(1)
 3000 FOR D%=0 TO 3:GCOL 0,15
 3010 READ J%,K%,M%,Q%,R%,S%
 3020 A%=-1:IF D%=0 OR D%=3 A%=1
 3030 B%=1:IF D%=1 OR D%=3 B%=-1
 3040 IF (d% AND 2^D%)=0 GOTO 3130
 3050 FOR N%=-8 TO 0 STEP 4
 3060 MOVE J%+N%*A%*2,K%-N%*B%:DRAW J%+N
%*A%*2,M%-N%*B%:DRAW Q%+N%*A%*2,R%-N%*B%
:DRAW Q%+N%*A%*2,S%-N%*B%
 3070 NEXT
 3080 GCOL 0,0
 3090 IFD%=0 IFP%=D%(L%,0) T%=D%(L%,1):G
COL0,T% EOR8
 3100 IF(D% MOD2)=0 AND (?&359=0 OR ?&35
9=192) GOTO3130
 3110 J%=J%-A%(D%,0):K%=K%-A%(D%,1)
 3120 MOVE J%,K%:MOVE J%,M%-A%(D%,2):PLO
T85,Q%-A%(D%,3),R%-A%(D%,4):MOVE Q%-A%(D
%,3),S%-A%(D%,5):PLOT85,J%,K%
 3130 NEXT
 3140 V%=X%:W%=Y%:U%=Z%:D%=2:F%=0
 3150 GCOL3,8:PROCdr(X%,Y%,Z%)
 3160 S%=0:FOR N%=1 TO 13
 3170 IFN%(N%,1)=P% S%=N%:PROCobj(S%)
 3180 NEXT
 3190 A%=-1:IFG%?P% A%=RND(5)+1:B%=RND(5
)+1:PROCplot(A%,B%,236)
 3200 ENDPROC
 3210 :
 3220 DEFPROCpad(C%)
 3230 GCOL0,0:MOVE256,444
 3240 MOVE320,412:PLOT85,256,380
 3250 MOVE192,412:PLOT85,256,444
 3260 GCOL0,C%:MOVE256,444
 3270 DRAW320,412:DRAW256,380
 3280 DRAW192,412:DRAW256,444
```

```
3290 ENDPROC
3300 :
3310 DEFPROCinit
3320 a%=&900:b%=&960:r%=&9C0:n%=&A02
3330 G%=&AB0
3340 DIM A%(3,5),D%(3,1),N%(13,2),T%(3,
1),R%(3)
3350 RESTORE 3840
3360 FOR D%=0 TO 3:FOR N%=0 TO 5
3370 READ A%(D%,N%):NEXT,
3380 FOR N%=0 TO4:READ $(n%+N%*32):NEXT
3390 FOR N%=0 TO 3
3400 READ T%(N%,0),T%(N%,1):NEXT
3410 FOR N%=0 TO 3:READ R%(N%):NEXT
3420 N%=0:FOR Y%=0 TO 8:A%=Y%+8:B%=Y%
3430 FOR X%=N% TO N%+8:a%?X%=A%:A%=A%-1
3440 b%?X%=B%:B%=B%+1:NEXT:N%=N%+9:NEXT
3450 ENDPROC
3460 :
3470 DEFPROCchars
3480 VDU23,224,60,122,126,2,42,2,60,126
3490 VDU23,225,126,127,183,167,126,60;0
3500 VDU23,226,60,94,126,64,84,64,62,12
6
3510 VDU23,227,126,-2,237,229,126,60;0
3520 VDU23,228,0;24,60,60,24,0;
3530 VDU23,229,42,85,85,85,85,54,20,8
3540 VDU23,230,0;-1,161,225,-1,0;
3550 VDU23,230;64,160,191,213,191,160,6
4
3560 VDU23,231,-2,-2,236,-2,238,238,238
,-2
3570 VDU23,232,4,9,13,14,24,48,96,192
3580 VDU23,233;0;170,170;0;
3590 VDU23,233;0,229,191,159,165,192;
3600 VDU23,234,126,66,66,66,126,60,66,-
1
3610 VDU23,235,24,36,36,126,-1,86,126,1
26
3620 VDU23,236,60,126,221,201,-1,221,98
,60
3630 VDU23,237,60,126,126,62,62,62,60,1
26
3640 VDU23,238,126,127,-1,-1,126,60;0
3650 VDU23,239,60,126,126,124,124,124,6
2,126
3660 VDU23,240,126,-2,-1,-1,126,60;0
3670 ENDPROC
3680 :
3690 DEFPROCenvs
3700 ENVELOPE1,1,0,-4,0,0,50,0,42,-16,-
2,-2,126,94
3710 ENVELOPE2,133,8,4,8,3,1,1,126,0,0,
-10,80,0
3720 ENVELOPE3,1,16,-12,1,25,25,25,10,-
10,0,-10,80,25
3730 ENDPROC
3740 :
3750 DEFPROCtitle
3760 VDU5,23;10,32;0;0;0;
3770 PROCprint("POSIDROID",352,892,4,6)
3780 PROCprint("By Jonathan Temple",64,
796,1,3)
3790 TIME=0:REPEAT UNTIL TIME>300
3800 ENDPROC
3810 :
3820 DATA 0,4,4,10,3,4,10,21,85,1,12,0,
5,4,1,21,85,10,30,4,1,12,85,10,21,0,6,4,
10,30,85,19,21,4,10,21,85,19,12,0,4,4,10
,3,85,10,21,5,10,30
3830 DATA 864,236,428,992,492,300,288,5
24,716,416,780,588,416,236,428,288,492,3
00,864,588,780,992,716,524
3840 DATA -8,-4,0,16,12,8,-24,-12,8,8,2
4,4,0,4,4,0,4,4,-8,4,28,24,12,-12
3850 DATA "Recharging pack","Electronic
 pass key","The floppy disc","The spanne
r","The pulse laser"
3860 DATA 0,48,22,38,35,3,54,31,3,31,38
,48
3870 DATA 0,26,3,0,36,3,0,60,3,0,13,3,1
,19,1,1,40,2,1,30,3,1,14,6,2,39,1,3,25,6
,4,62,1,5,10,7,6,24,5
3880 DATA 6,3,29,2,42,1,57,6,4,6,8,15,1
6,18,27,29,32,34,41,46,50,57,59,61,6,1,2
,3,6,6,6,7
3890 :
3900 MODE7:REPORT:PRINT" at line ";ERL
3910 *FX 15,1
3920 END
```


Thus, to see if record "n" refers to a married woman:

```
1200 IF NOT FNReadBit(SMap%,n)
        AND FNReadBit(MMap%,n) THEN..
```

To investigate the effect of bit maps, I created a dummy file with 1000 50-byte records. The records included sex and married status fields, and roughly one in 16 identified a married woman. A conventional search of the file for those records took 120 Secs. The same search via bit maps, including recovering the records, took just 51 Secs. A search with a directly coded routine, rather than using EVAL, took 37 Secs. Setting up the 2 bit maps took 123 Secs (but only had to be done once, remember).

Obviously, avoid setting up the maps from scratch too often. Once they are formed, they are best saved on disc with a *SAVE. They can then be reloaded with *LOAD. If that's a new idea to you, I'm afraid that I'm out of space this month but, Editor willing, I will come back to it in the future.

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.

Editor: Mike Williams
Technical Assistant: Alan Webster
Production Assistant: Yolanda Turuelo
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft
Additional thanks are due to Sheridan Williams, Adrian Calcraft, Geoff Bains, John Yale and Tim Powys-Lybbe.

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Publications Limited.
BEEBUG Publications Ltd (c) 1986

Editorial Address

BEEBUG
PO BOX 50,
Holywell Hill,
St. Albans AL1 3YS

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 above. If you require a reply it is essential to quote your membership number and enclose an SAE.

MEMBERSHIP RATES

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

MEMBERSHIP SUBSCRIPTION RATES

£ 6.90 6 months (5 issues) UK ONLY
£12.90 - 1 year (10 issues) UK, Eire, BFPO
£19.00 - Rest of Europe £22.00 Middle East
£24.00 Americas & Africa, £26.00 Elsewhere

BACK ISSUES
(Members only)

Vol	Single issues	Volume sets (10 issues)
1	90p	£8
2	£1	£9
3	£1.20	£10
4	£1.20	£11
5	£1.30	–

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 requires 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.

Membership, Back Issues & Software Address

BEEBUG
PO BOX 109
St. Johns Road
High Wycombe HP10 8NP

Hotline for queries and software orders

St. Albans (0727) 40303
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.

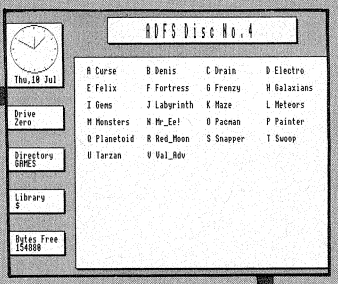
Magazine Cassette/Disc

CASSETTE/DISC CONTENTS AUG/SEPT 1986

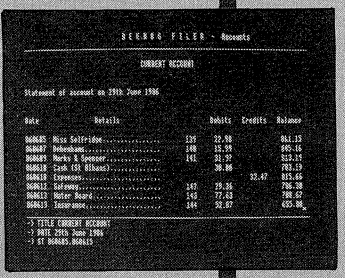
DISC MANAGER — the utility that really does save time when managing your disc files.
MAPPING THE BRITISH ISLES — map out the British Isles in any scale and position.
THE MASTER SERIES — a stunning ADFS menu using windows and icons for an impressive display.
PAINTINGS BY NUMBERS (PART 2) — another demonstration of this fast fill routine.
FILER ACCOUNTS OPTION — turns the BEEBUG database into a home banking system that is simplicity itself to use.
INSTANT TEXTURING — for a professional yet easy to produce background to your displays.
FIRST COURSE — examples of the GET function, including a useful menu routine.
SOFTWARE FOR SIDEWAYS RAM (Part 3) — final selection of utilities in this useful series.
POSIDROID — nothing less than an enthralling and challenging 3D adventure game.

EXTRA FEATURES THIS MONTH
MAGSCAN — data for this issue of BEEBUG (Vol. 5 No. 4).

ADFS Menu



Filer Accounts





Mapping Britain

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
6 months (5 issues)	£25.50	£17	£30
12 months (10 issues)	£50	£33	£56

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

Cassette subscriptions can be commuted to disc subscriptions at a cost of £1.70 per issue of the subscription left to run.

All subscription and individual orders to:
BEEBUG, PO BOX 109, St. Johns Road, High Wycombe, HP10 8NP

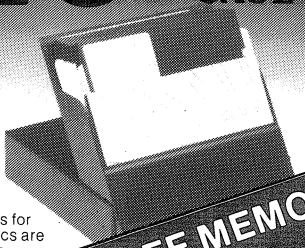
See special offer on subscription rates with this issue.

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 pla… libr…

25 **FREE STORA…**



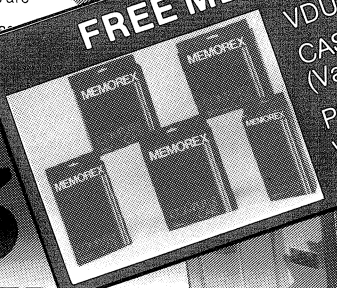
FREE MEMOREX CLEANING KITS

VDU/TV Kit with 10 discs (Value £4.65)

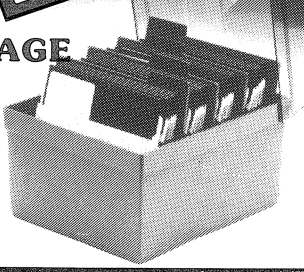
CASE/KEYBOARD & VDU/TV with 25 discs (Value £9.30)

PRINTER, VDU/TV & CASE/KEYBOARD with 50 discs (Value £17.25)

Only with the voucher below



5 **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	£13.90	10	D/S	D/D	£18.90
25	S/S	D/D	£33.50	25	D/S	D/D	£43.90
50	S/S	D/D	£56.90	50	D/S	D/D	£77.90
96 TPI DOUBLE DENSITY							
10	S/S	D/D	£18.90	10	D/S	D/D	£19.90
25	S/S	D/D	£43.90	25	D/S	D/D	£46.20
50	S/S	D/D	£77.90	50	D/S	D/D	£84.90

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.

ORDER FORM

Official orders welcome
Access/Barclaycard
24 hrs line 0494 816666
Further information on helpline 0727 40303

Please supply me

@ £

@ £

@ £

Total cheque enc. £

Name

Address

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

MEMOREX COMPUTER CARE KITS OFFER VOUCHER