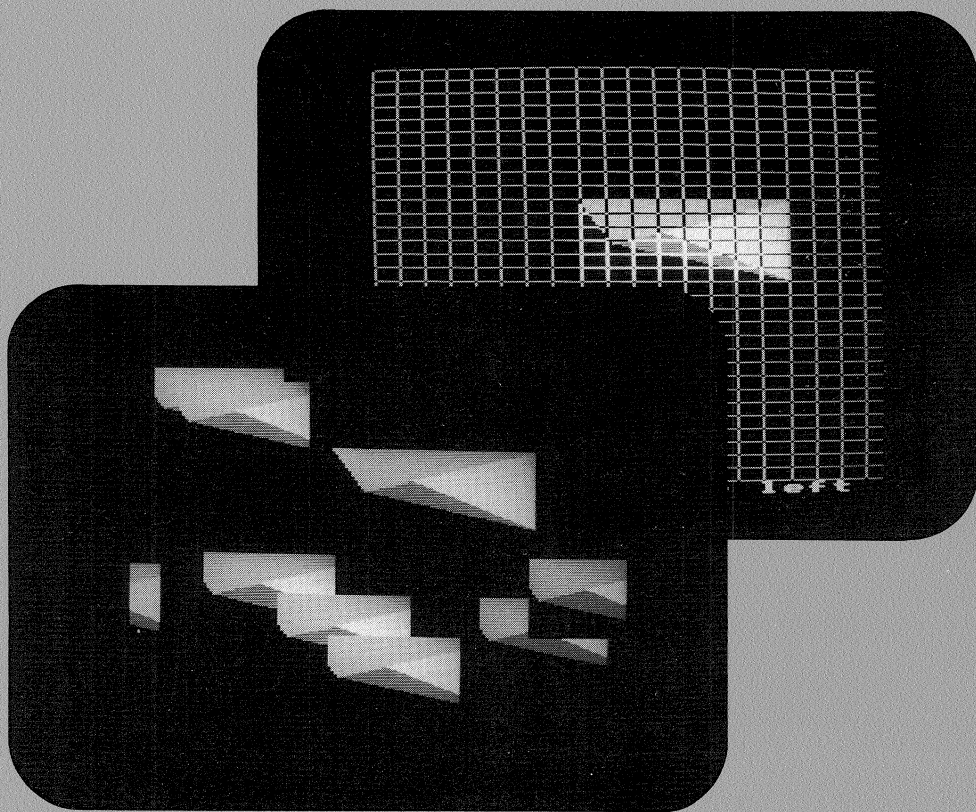


VOLUME 5 ISSUE 9

MARCH 1987 PRICE £1.30

BEEBUG

FOR THE BBC MICRO



GRAPHICS BLOCK MOVE

MASTER
AUTO ALARM

BEERNA FLYER - Accounts

90401	Safeway	137	32.56	946.46
90402	Freemover	138	12.27	594.13
90403	Cash (Kitchin)		56.00	889.13
90404	Miss Selfridge	139	22.50	964.13
90405	Hubbards	140	15.50	979.13
90406	Harks & Spencer	141	11.97	991.13
90407	Cash (St Hilman)		38.00	1029.13
90408	Lynsons			1055.66
90409	Safeway	142	25.35	1081.00
90410	Water Board	143	77.63	1158.63
90411	Insurance	144	52.37	1211.00
90412	British Telecom	145	50.37	1261.37
90413	BTCL	146	9.00	1270.37
90414	Bovis	147	15.54	1285.91
90415	Clarks Camera Centre	148	139.25	1425.16
90416	Dents	149	9.03	1434.19

> ST 90409
 Balance at 90416 is 944.71
 > *

Daily Max and Min temperatures - Barton, beds.
January 1986

1985
 1986
 1987

Volume 5 Number 9
March 1987

New Fonts for your Beeb	7
Graphics Block Move	10
BEEBUG Education	14
Multi-Character Printer Driver for View	17
New OS calls Uncovered	21
Virtual Arrays (Part 2)	25
The Master Series	
Master Add-ons Review	37
A Versatile Auto-Alarm System	39
Master Hints	42
File Database System — The Last Word	43
BEEBUG Workshop — Recursion	46
First Course	
Making Good Use of Errors Part 2	54
Labyrinth	57

Cumana's Astron Card	5
New Chess Grandmaster for the Beeb?	12
The ADE+ Assembler System	23
Maximising your Mouse	48
Watford Drives View	50

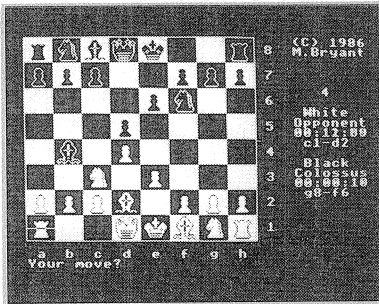
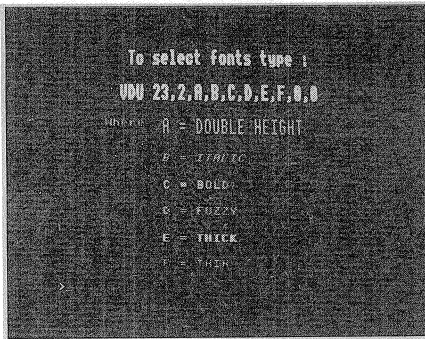
Editorial Jottings	3
News	4
Supplement	29-36
Hints and Tips	52
Postbag	53

GENERAL

Preserving Tube Test
Clearing a REPEAT Loop
FX80 Multi-Font Printout
Zero Page Workspace
Basic Boots
1770 Free Gift
Double Blurr

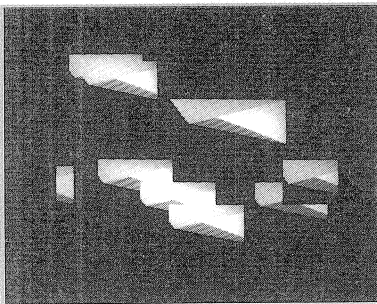
Don't Nest too Deep
Use Wild Cards
Using Function Keys
Function Key Doubling
Extra key Strips

New Fonts



Colossus 4 Chess

Graphics Block Move



ON	ESC Function	OFF
---*A	UNDERSCORE	---*@
-E	EMPHASIZED	-F
-G	DOUBLE STRIKE	-H
--W*a	WIDE	--W*'
--S*@	SUPERSCRIPT	-T
--SXA	SUBSCRIPT	-T

ON	CTRL Function	OFF
*N	WIDE	*T
*O	CONDENSED	*R
*G	BELL !	

View Printer Driver

ACORN SUPPORT

Acorn have announced changes to their customer support role, to take effect by the end of March. In future, Acorn will not provide direct support to end users, but will expect authorised dealers to do this. Acorn's own support will be revamped to provide fast and efficient backup to dealers. One major feature of the new service will be a considerable reliance on a viewdata service, which will provide down-loading of information and software. Large parts of this will be accessible directly by users with the necessary equipment, though some parts will be for dealers only.

BEEBUG is a full Acorn dealer, so we will have the benefit of Acorn's support as before, and together with our own Customer Support department, we can offer help with most aspects of using your BBC Micro, Master or Compact.

BEEBUG MAGAZINE OFFERS

We can still supply the BEEBUG Sideways RAM module (see BEEBUG Vol. 5 No.7) either as a kit for you to make up yourself, or as a ready made unit. This is a really cheap way to add sideways RAM to your machine. With this issue, David Graham describes an Alarm System for Master owners, and this is available as a ROM for just £6.45. We have also produced a disc containing all the Filer programs, updated and improved as described in this issue, with a revised and extended set of notes. These members' offers are described more fully in this month's supplement.

BEEBUG ON MICRONET

As announced last month, we are establishing a major database on Micronet. Although this is likely to be incomplete for a while yet, Micronet browsers may like to take a peek at *800909#.

PROGRAM/REVIEW CLASSIFICATION

We hope that the new classification symbols for programs and reviews clarify matters with regard to the variety of Acorn systems. The complete set of icons is shown below. These show clearly the valid combination of machine (version of Basic) and filing system for each item, the Tube compatibility. A single line through a symbol indicates partial working (normally just a few changes will be needed); a cross shows total incompatibility. Reviews do not distinguish between Basic I and II.

COMPUTER SYSTEM

Master (Basic IV)

Compact (Basic IV)

Compact (Basic VI)

Model B (Basic II)

Model B (Basic I)

Electron

FILING SYSTEM

ADFS

DFS

Cassette

TUBE COMPATIBILITY

Tube

News News News News News News Ne

Quack Quack

Now available for Fleet Street Editors is a disc full of Walt Disney cartoon characters such as Mickey Mouse and Donald Duck. And there's no copyright restriction as long as the resultant work is not sold for profit.

Also from Mirrorsoft comes Admin Xtra, four utilities for use with Fleet Street Editor; Mode Conversion, Disc Indexing and Filing, Poster Maker and Panel Viewer.

Walt Disney Graphics and Admin Xtra both cost £14.95 on either 40 track or 80 track disc. Mirrorsoft are on 01-377 4837.

Puffer Trains etc

William Poundstone's 'The Recursive Universe' is described as a "fantastic explosion of intellectual fireworks". The book offers a wide ranging discourse on life and the universe based on the game of Life devised by John Conway. You can learn about gliders, puffer trains, beehives and many more fascinating and changing life-forms, and a listing is included to implement Life on any micro. The book costs £5.95 in paperback and is published by Oxford University Press. You can contact them on (0865) 56767.

Real Graphics

New from the author of Glentop's 3D Graphics System (see BEEBUG Vol.4 No.5) is the Real Time Graphics System from Silicon Vision. This package allows you to create 3D wireframe objects which can be accessed from your own Basic or Assembler programs to produce stunning animated sequences.

The system is supplied as a 32K Graphics ROM plus five discs and a 115 page manual all for £79.95. Ideal for creating your own Elite style games, and for many more serious applications. Contact Silicon Vision on 01-422 2274.

Help for Teachers

For teachers bewildered by the volume and variety of software available for schools, BBC Education has produced an evaluation pack. This contains three discs: one with samples of primary software, one with samples of secondary software, and the third providing complete demonstrations of 'Maths with a Story' and 'Picture Craft'. All of these are from educational software produced by the BBC, but the packs are being distributed free through LEA advisers. Additional copies cost £5.00 can be had from David Watkins, Education Officer, Computer Software, BBC, Villiers House, The Broad-

way, London W5 2PA, or telephone 01-743 8000.

Warbling Red Boxes

Red Boxes (see review in BEEBUG Vol.5 No.7) can now 'warble'. Known as Red Five, this is the latest addition to the Red Box home security system that can be based on a BBC micro. The new unit offers 160db of warble, and costs £36.95 inc post & packing. A project manual is also available at £9.95 to help you make the most of your Red Boxes which are supplied by Electronic Fulfilment Services Ltd. on (0223) 323143. They may also be purchased through BEEBUG with our normal 5% discount for members.

Sci-speak

If English is insufficient for your word processing needs then the SCIWAYS ROM from Mayhew Telonics may be what you need. This allows easy printing of science-based characters and symbols, both on the screen and on Epson-compatible printers. The entire Greek character set is supported, as are all the main scientific and mathematical symbols, subscript, superscript and user-defined characters. SCIWAYS can be used with either Wordwise or View, and costs £38.53 inc VAT. Mayhew may be found on (0202) 747695.

B

CUMANA'S Astron Card

Is the Astron Card the next major storage revolution or a nine days wonder? Simon Williams has been giving the question some thought.

Product : Astron card adaptor, boxed socket and sample game on card

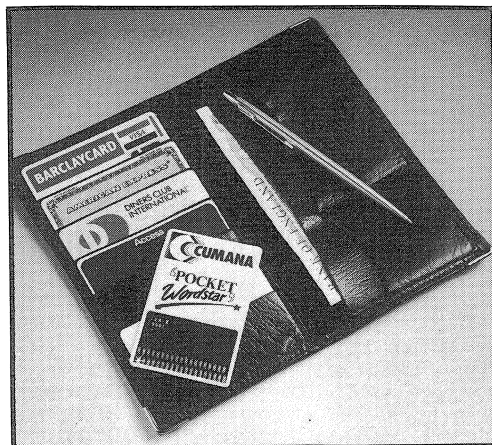
Supplier: Cumana Ltd.
Pines Trading Estate,
Broad Street,
Guildford, Surrey GU3 3BH.
Tel. (0483) 503121

Price : £24.95 (BBC micro),
£44.95 (Master)

In the last ten years or so, small rectangles of plastic have made larger and larger inroads into our daily lives. Credit cards from banks have increased the convenience of shopping, those from individual stores have offered instant credit to spread the cost of purchases. The information these cards carry is embossed in the plastic itself and read by running an inked roller across the raised type.

The banks have gone further than this with the introduction of automatic cash dispensers. The information on cash cards is encoded into magnetic strips across their backs. The dispenser can read this code and verify the card before allowing the user to withdraw money from the machine at any time.

Even with the autobank card, though, the amount of information which can be stored is relatively small. The latest evolution of the plastic card offers far greater potential, and may well prove to be the most important computer innovation since the advent of the floppy disc. In this country, one of the leading exponents of 'chip-on-a-card' technology is the disc drive manufacturer, Cumana, and you can try out 'Astron' cards, as the company



calls them, with an inexpensive adaptor for your BBC Micro or Master.

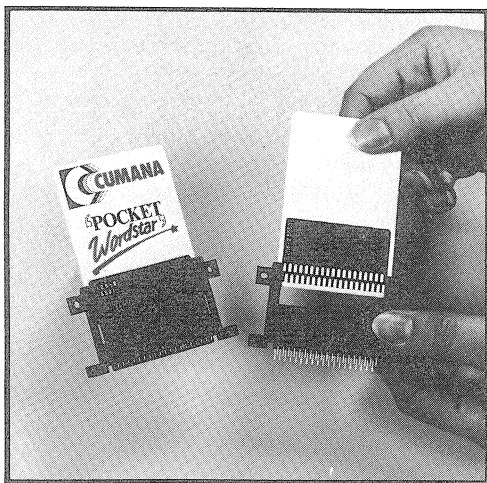
The BBC Micro adaptor consists of a small plastic box connected to the micro by a half metre length of ribbon cable. The adaptor plugs into any vacant ROM socket, and can be used from an expansion board as well as from the main p.c.b. The cable has to come out between the case halves or through the ventilation slot at the rear. It's a shame an insert couldn't be made for an Astron edge connector to fit in the speech ROM hole to the left of the keyboard.

Inside the box there is a small circuit board and the card socket itself, which is similar to an edge connector, but with guides at either end to ease the insertion of the card. The adaptor for the Master fits into the cartridge port, and is rather more expensive than the BBC unit.

The Astron card is a credit card-sized piece of 2mm thick plastic, with two rows of gold-plated contacts at one end. Embedded in the plastic is a ROM, RAM or EPROM chip, much like the chips inside 'normal' 28 pin ROM packages. The card is very durable, and quite tough enough to slip in a wallet or pocket. The system could be further miniaturised, as the chip is mounted close to the contacts, and a lot of the length of the card is there to take advertising or instructions.

Once the adaptor has been connected to the micro (with the power switched off), you insert the card and switch on. No matter which socket the adaptor is plugged into, it takes priority and the title screen for the supplied Superior Software game comes up on your screen. To be a direct replacement for a disc, you should really be able to insert and extract cards with your micro on, and to select them with the ROM filing system. Perhaps this will come later.

The game itself, 'Star Striker', is a copy of the old arcade favourite 'Moon Cresta', where you plough your way through wave after wave of aliens with a three tier ship that is forever separating and rejoining. If anything, the game is rather too easy to play, and once you've been through half a dozen waves they start to repeat. The game does serve to point out the advantages of the Astron card, though: instant loading, no disc errors and the possibility of holding many kilobytes in your pocket.



So far, the Astron card has moved very slowly in gaining acceptance from software houses and the public. It's a bit of a 'chicken and egg' problem, as the average micro-user will only show passing interest while there is a shortage of software in the new medium, and software producers are reluctant to

invest in another form of data storage while there are few potential buyers.

The two main suppliers to have expressed interest so far are Superior Software, who are to market several of their BBC Micro games on Astron cards at the same price as the disc version, and ACP, who can provide a selection of their utilities 'on-card'.

The drawback at the moment is the price of the cards themselves. Although 8K and 16K EPROM cards are available for between £5 and £10, this compares unfavourably with the 28-pin chip variety. The advantages of portability and robustness have to be weighed against the extra cost. Perhaps for this reason, Cumana are concentrating on the serious software market for their cards. With business and utility software, the extra cost of the card represents a smaller proportion of the total. The company is also looking at other target micros, and there are rumours that the portable machine from Clive Sinclair's new company may make use of them.

Perhaps the most exciting feature of the Astron card is the development of large capacity ROM and RAM cards. Cards of up to half a Megabyte are already available (at a price) and 1 and 2 Megabyte ones are expected soon. Imagine carrying the equivalent of a hard disc full of data on five credit cards! RAM cards will be battery-backed with a five year lithium cell and Cumana are developing a DFS-like filing system to allow RAM cards to be used as discs.

At present, the price of the cards probably restricts them to EPROM chip replacements, where their added convenience and portability are assets. If Cumana can get enough adaptors onto the market, as they deserve to, to be able to offset the development costs of the larger ROM and RAM cards, the Astron card could take over as a primary means of data storage.

The reasonable cost of the adaptor gives you the chance to try the new technology for yourself. And you can always zap a few aliens while you're at it.

B

NEW FONTS FOR YOUR BEEB

Tired and bored with just one style of lettering on your screen? Thomas Williamson's program provides a much better choice.

It is often useful to be able to display text in a variety of different styles. This program allows you to do so easily with a simple VDU command. It provides six new fonts, and any number of these can be combined to provide further interesting effects. The fonts are:

1. Double height
2. Italics
3. Bold
4. Fuzzy
5. Thick
6. Thin

USING THE PROGRAM

Type the program into your computer and save it to disc or tape. Then run the program to assemble the machine code routine needed, and you are ready to choose a new style of lettering.

To select any font you can now use a VDU command in the form:

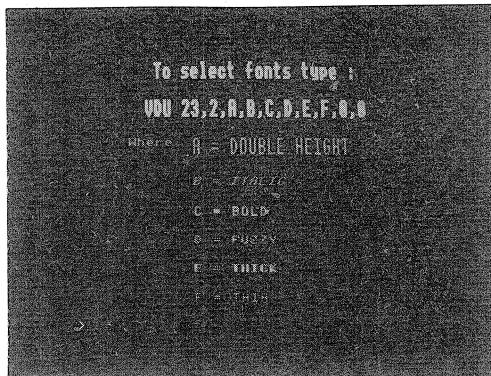
```
VDU23,2,f1,f2,f3,f4,f5,f6,0,0
```

The value of f1 determines the state of font 1 (double height), f2 the state of font 2 and so on. The values to use for each 'f' are:

- 1 to switch a font on
- 0 to switch a font off

while any other value will leave a font as it was (in the examples we have used a '9'). The two zeros must always be included at the end, as a VDU 23,2 command always expects eight values to follow.

Once a font (or combination of fonts) has been selected, all text will be



displayed in that font, except in mode 7.

To obtain double height fuzzy text, for example, type:

```
VDU23,2,1,0,0,1,0,0,0,0
```

To set italics without affecting the other fonts, type:

```
VDU23,2,9,1,9,9,9,0,0
```

Note, that even if you are changing only one of the fonts, all eight numbers should still be included. The VDU 23,2 command may be used either in immediate mode, or in your own programs, providing the code to implement this has been activated first.

If you want to save the machine code so that it can be *RUN, run the program to assemble the code as before, and then type:

```
*SAVE FONTMC 960 +140
```

You can, of course, replace FONTMC by any name of your choice. If you set a font before saving the code, then that font will be automatically selected when the code is *RUN. If you write a program using the extended VDU 23 calls to select different fonts, include *RUN FONTMC near the start of the program so that the routine is loaded automatically.

Note that characters 224 and 225 are used by the program to create the new fonts, so these should not be redefined while the program is in operation.

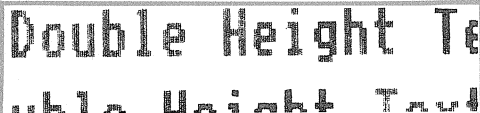
HOW THE PROGRAM WORKS

Two vectors are used by this program. The VDU extension vector at &226/&227 is used to create the new VDU command. On entry the carry flag is set to indicate that an unrecognised VDU 23,n has been issued. The accumulator holds 'n' and the eight parameters are stored in locations &31C to &323.

The program also uses the write character vector at &20E/&20F to intercept characters before they are output. Then, having checked the character range, the VDU queue, and the current mode, it puts the characters definition into &C00-&C07 and alters it accordingly before printing it. This technique was covered in the two 'First Course' articles in Vol.5 Nos. 6 & 7, and you should refer to these articles if you require more information.


USING A PRINTER

Although the character definitions are altered for display on the screen, any output to a printer will consist of a stream of ASCII codes which will be printed normally. If you wish to print these new characters, then a suitable graphics screen dump program must be used.



```
10 REM Program New Fonts
20 REM Version B0.1
30 REM Author T.Williamson
40 REM BEEBUG March 1987
50 REM Program subject to copyright
60 :
100 ON ERROR GOTO 170
110 MODE 1
120 PROCfonts
130 CALL start
140 PROCdemo
150 END
160 :
170 ON ERROR OFF
180 MODE 7
190 IF ERR=17 THEN END
200 REPORT:PRINT" at line ";ERL
210 END
220 :
1000 DEFPROCdemo
1010 VDU23,2,1,0,1,0,1,0,0,0
1020 PRINTSPC9"To select fonts type :"
```

```
1030 PRINT'SPC8"VDU 23,2,A,B,C,D,E,F,0,0"
1040 VDU23,2;0;0;0
1050 COLOUR2:PRINTSPC6"Where";
1060 FORA%=0TO5
1070 READA$
1080 VDU23,2
1090 IFA%>0FORB%=1TOA%:VDU0:NEXT
1100 VDU1:FORB%=1TO7-A%:VDU0:NEXT
1110 PRINTTAB(13)CHR$(65+A%) " = "A$'
1120 IFA%>0 PRINT
1130 NEXT
1140 VDU23,2;0;0;0;0
1150 COLOUR3
```



```
1160 ENDPROC
1170 :
1180 DATADouble HEIGHT,ITALIC,BOLD,FUZZ
Y,THICK,THIN
1190 :
1200 DEFPROCfonts
1210 oldvdu=!(((!FFB7 AND&FFFF)+&26)
1220 oldwrch=!(((!FFB7 AND&FFFF)+&E)
1230 O%=oldwrch
1240 osword=&FFF1
1250 mem%=&900
1260 FORI%=0TO2STEP2
1270 P%=mem%:data=P%
1280 !P%=0:P%!=0
1290 P%=P%+8
1300 [OPTI%
1310 .newvdu
1320 PHP
1330 BCC notvdu
1340 CMP#2:BNE notvdu
1350 PHA:TXA:PHA
1360 LDX#7
1370 .move
1380 LDA&31C,X
1390 AND#1:CMP&31C,X
1400 BNE nostore
```



```
1410 STA data,X
1420 .nostore
1430 DEX:BPL move
1440 PLA:TXA:PLA:PLP
1450 RTS
1460 .notvdu
1470 PLP:JMP oldvdu
```



```

1480 :
1490 .newwrch
1500 PHP:PHA
1510 STA&BFF
1520 LDA&356:CMPI#4
1530 BEQ exit
1540 LDA data:BEQ notdbl
1550 PLA:PHA

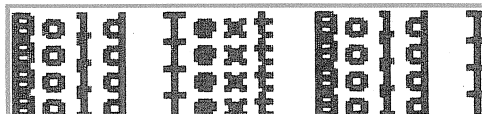
```



```

1560 CMPI#13:BNE del
1570 LDA#10:JSR 0%
1580 JMP exit
1590 .del
1600 CMPI#127:BNE notdbl
1610 LDA#10:JSR 0%
1620 LDA#127:JSR 0%
1630 LDA#11:JSR 0%
1640 LDA#9:JSR 0%
1650 JMP exit
1660 .notdbl
1670 PLA:PHA
1680 SEC:SBC#32:BMI exit
1690 SEC:SBC#95:BPL exit
1700 TXA:PHA:TYA:PHA
1710 LDX#&FF:LDY#&B
1720 LDA#&A
1730 JSR osword
1740 LDA#0:STA&C08
1750 JSR choices
1760 LDA#224:JSR 0%
1770 LDA data:BEQ nrml
1780 LDA#10:JSR0%:LDA#8:JSR 0%
1790 LDA#225:JSR 0%:LDA#11:JSR 0%
1800 .nrml
1810 PLA:TAY:PLA:TAX:PLA:PLP
1820 RTS
1830 :
1840 .exit
1850 PLA:PLP

```



```

1860 JMP oldwrch
1870 :
1880 .choices
1890 LDA data+1:BEQ nxt1
1900 LSR&C00:LSR&C01
1910 ASL&C05:ASL&C06:ASL&C07
1920 .nxt1
1930 LDA data:BEQ nxt2

```

```

1940 LDX#7:LDY#14
1950 .dbl
1960 LDA&C00,X
1970 STA&C00,Y
1980 STA&C01,Y
1990 DEY:DEY:DEX
2000 BPL dbl
2010 .nxt2
2020 LDA data+2:BEQ nxt3
2030 LDX#14
2040 .bld
2050 LDA&C01,X
2060 ORA&C00,X
2070 STA&C01,X
2080 DEX:BPL bld
2090 .nxt3
2100 LDA data+3:BEQ nxt4
2110 LDX#15
2120 .fzy
2130 LDA&C00,X
2140 ASLA
2150 STA&C00,X
2160 DEX:DEX
2170 BPL fzy

```



```

2180 .nxt4
2190 LDA data+4:BEQ nxt5
2200 LDX#15
2210 .thk
2220 LDA&C00,X
2230 ASLA:ORA&C00,X
2240 STA&C00,X
2250 DEX:BPL thk
2260 .nxt5
2270 LDA data+5:BEQ done
2280 LDX#15
2290 .thn
2300 LDA&C00,X
2310 ASLA:AND&C00,X
2320 STA&C00,X
2330 DEX:BPL thn
2340 .done
2350 RTS
2360 :
2370 .start
2380 LDX#newwrch MOD256
2390 LDY#newwrch DIV256
2400 STX&20E:STY&20F
2410 LDX#newvdu MOD256
2420 LDY#newvdu DIV256
2430 STX&226:STY&227
2440 RTS
2450 ]NEXT
2460 ENDPROC

```

GRAPHICS BLOCK MOVE

Nicholas Brozovic's useful technique allows any part of a graphics screen to be copied to another part of the screen. Use it in its own right, or incorporate it in your own graphics programs.

The program listed here will allow any part of a mode 2 graphics screen to be copied to any other part of the screen. The basic procedure can be incorporated into your own graphics programs, or you may use the complete program given here.

This allows any mode 2 screen to be loaded from disc (or tape). A grid is then displayed on the screen and a rectangular area marked out to be copied. A new position is then marked and the part previously selected copied to this position. The process may be repeated as many times as you wish and the results eventually saved.

USING THE PROGRAM

Once the program has been typed in and saved, you can try running it. A sample graphics screen is displayed, but you can replace this as prompted by loading an alternative graphics screen from tape or disc, or by replacing the drawing procedure with one of your own.

Then a grid is displayed, and the 'z', 'x', ':' and '/' keys used to move a cursor left, right, up and down respectively. Each position is 'fixed' by pressing the space bar (at top left, top right and bottom left). Once confirmed, the cursor is similarly moved to the top left hand corner of the intended image position, and the space bar pressed to produce the copy.

The same selected area can be readily copied into many different positions, producing quite interesting effects, and the results saved as and when required.

UNDERSTANDING THE PROGRAM

The actual copy procedure (PROCmap) has two parameters Y2% and Y%. These are the start position to copy from, and the start position to copy to, both screen addresses. The procedure uses two other variables (again screen addresses) that must be set beforehand; ENDP0S which references the top right corner, and ENDBOT which references the bottom left. For more information on how to relate graphics to screen addresses, see the series 'Machine Code Graphics' Vol.2 Nos.8, 9 & 10, Vol.3 Nos.1 & 2.

The other procedures used in the demo program are as follows:

demo	main demonstration
draw	sample graphics drawing routine
loop	main loop for repeated copy
coord	collects co-ordinates of points
grid	used to display or remove grid
loc	looks after cursor movement
choice	offers choice of load or save
save	save screen
load	load screen
message	display messages
ls	display load/save message
filename	get filename
oscli	perform OSCLI function

Graphics programs have always been popular with many Beeb users, and whether you just use the basic procedure, or the entire program, this routine should prove most useful.

```

10 REM Program Block Move
20 REM Version B1.2
30 REM Author Nicholas Brozovic
40 REM BEEBUG March 1987
50 REM Program subject to copyright
60 :
100 ON ERROR GOTO 150
110 MODE 2
120 PROCdemo
130 END
140 :
150 ON ERROR OFF
160 MODE 7
170 IF ERR=17 END
180 REPORT:PRINT" at line ";ERL
190 END
200 :
1000 DEFPROCdemo
1010 VDU5:VDU23,225,24,24,24,255,255,24
,24,24
1020 F=&1EF:G=&90

```

```

1030 move=FALSE:quit=FALSE
1040 X2%=&0:X%=&7700
1050 VDU23,1,0;0;0;0;
1060 PROCdraw
1070 MOVE350,30:GCOL3,6:PRINT"Load (Y/N
)?":A$=GET$
1080 MOVE350,30:PRINT"Load (Y/N)?":IFA$
="Y" THEN PROCload
1090 PROCloop
1100 ENDPROC
1110 :
1120 DEFPROCdraw
1130 MOVE520,518:MOVE1038,518:GCOL0,1
1140 PLOT85,520,700:MOVE1038,700
1150 GCOL0,3:PLOT85,1038,518
1160 GCOL0,5:MOVE600,600:PLOT85,520,700
1170 GCOL0,4:MOVE1038,700:PLOT85,600,60
0
1180 ENDPROC
1190 :
1200 DEFPROCloop
1210 PROCgrid:PROCcoord
1220 REPEAT
1230 PROCmessage:move=TRUE
1240 PROCloc:PROCmessage
1250 IF quit=TRUE THEN PROCchoice:END
1260 PROCgrid:PROCmap(X2%,X%)
1270 PROCgrid
1280 UNTIL FALSE
1290 ENDPROC
1300 :
1310 DEFPROCcoord
1320 move=FALSE
1330 GCOL3,7:MOVE10,30:PRINT"Define : "
1340 MOVE700,30:PRINT"Top left"
1350 PROCloc:F1=F+25:G1=G-16:X2%=X%
1360 MOVE700,30:PRINT"Top left":MOVE650
,30:PRINT"Top right":PROCloc
1370 F2=F+25:G2=G-16:GCOL3,7:MOVEF1,G1:
DRAWF2,G2:ENDPOS=X%+7
1380 MOVE650,30:PRINT"Top right":MOVE57
0,30:PRINT"Bottom left"
1390 PROCloc:ENDBOT=X%-&280
1400 F3=F+25:G3=G-16:MOVEF2,G2:DRAWF2,G
3:DRAWF3,G3:DRAWF1,G1
1410 MOVE10,30:PRINT"Define :":MOVE570,
30:PRINT"Bottom left"
1420 IF G1<G2 OR F1<F3 THEN SOUND1,-1
5,100,7:GCOL3,7:MOVEF1,G1:DRAWF2,G2:DRA
W2,G3:DRAWF3,G3:DRAWF1,G1:PROCcoord
1430 MOVE350,30:GCOL3,6:PRINT"O.K. (Y/N
)?":A$=GET$
1440 MOVE350,30:PRINT"O.K. (Y/N)?":GCOL
3,7:MOVEF1,G1:DRAWF2,G2:DRAWF2,G3:DRAWF
3,G3:DRAWF1,G1
1450 IF A$="N" THEN PROCcoord
1460 ENDPROC
1470 :
1480 DEFPROCgrid
1490 GCOL3,2:FORX=&20TO1100STEP32

```

```

1500 MOVE0,X:DRAW1280,X:NEXT
1510 MOVE0,1023:DRAW1279,1023
1520 FORY=0TO1300STEP&41
1530 MOVEX,&20:DRAWY,1024:NEXT
1540 ENDPROC
1550 :
1560 DEFPROCmap(Y2%,Y%)
1570 FORA=0TO (ENDPOS-Y2%)
1580 FORB=0TO (ENDBOT-Y2%) STEP&280
1590 ?(A+B+Y%)=? (A+B+Y2%)
1600 NEXT:NEXT
1610 ENDPROC
1620 :
1630 DEFPROCloc
1640 GCOL3,7:REPEAT
1650 MOVEF,G:VDU225:A$=GET$
1660 MOVEF,G:VDU225
1670 IF A$="Z" THEN F=F-&41:X%=X%-&20
1680 IF A$="X" THEN F=F+&41:X%=X%+&20
1690 IF A$="." THEN G=G+&20:X%=X%-&280
1700 IF A$="/" THEN G=G-&20:X%=X%+&280
1710 IFF<-&21 THEN F=&4BA:X%=X%+&280
1720 IFF>&4C5 THEN F=-&19:X%=X%-&280
1730 IFG>&410 THEN G=&30:X%=X%+&5000
1740 IFG<&1A THEN G=&410:X%=X%-&5000
1750 IF A$="Q" AND move=TRUE THEN PROCg
rid:quit=TRUE:ENDPROC
1760 UNTIL A$=" " OR A$="C"
1770 IF A$="C" AND move=TRUE THEN GCOL3
,7:MOVE575,30:PRINT"Move":MOVE0,30:GCOL3
,6:PRINT"C-Change":MOVE900,30:PRINT"Q-Qu
it":PROCcoord:GOTO1220
1780 *FX15,1
1790 ENDPROC
1800 :
1810 DEFPROCchoice
1820 GCOL3,6:PROCls:A$=GET$:PROCls
1830 IF A$="S" THEN PROCsave
1840 IF A$="L" THEN PROCload
1850 ENDPROC
1860 :
1870 DEFPROCsave
1880 PROCfilename
1890 CL$="SAVE "+fn$+" 3000 8000"
1900 PROCoscli(CL$)
1910 MOVE400,30:PRINT"Quit (Y/N)?"
1920 A$=GET$:IF A$="N" THEN MOVE400,30:
PRINT"Quit (Y/N)?:PROCloop
1930 ENDPROC
1940 :
1950 DEFPROCload
1960 PROCfilename
1970 CL$="LOAD "+fn$:PROCoscli(CL$)
1980 PROCloop
1990 ENDPROC
2000 :
2010 DEFPROCmessage
2020 GCOL3,7:MOVE575,30:PRINT"Move"
2030 GCOL3,6:MOVE0,30:PRINT"C-Change"
2040 MOVE900,30:PRINT"Q-Quit"

```


NEW CHESS GRANDMASTER FOR THE BEEB?



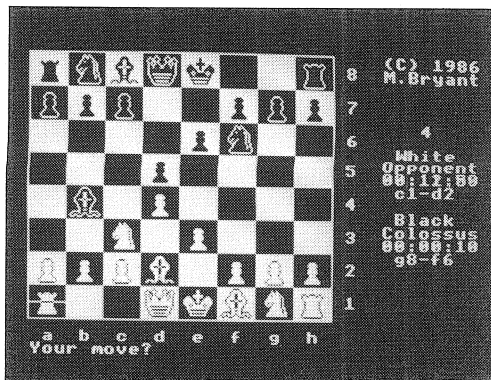
BBC Soft's White Knight has been the undisputed chess champion on the Beeb. Now a new challenger has arrived on the scene in the form of Colossus Chess 4.0. Bernard Hill, no mean player himself, has been refereeing the match.

Product : Colossus 4 Chess
CDS Software Ltd
Silver House, Silver Street,
Doncaster DN1 1HL.
Tel. (0302) 21134

Price : £14.95 (disc), £9.95 (cassette)
inc VAT and p&p.

Any new chess program for the Beeb written by Martin Bryant must be worth looking at. In Computer Chess circles, Martin's nine years' experience makes him quite an old timer, but unlike many chess programmers he has put his efforts into the home computer market. Previous versions of Colossus are already well known to owners of other computers, and Martin's previous BBC programs were White Knight Mk 11 and 12 published by BBC Soft. Colossus is available on tape or disc (40 track or 80 track - state which) for the Electron, BBC, B+ and Master. A Master Compact version will be available shortly, and disc versions contain programs for Model B and an expanded version for B+ or Master. The existence of both versions on one disc means you have no problems if you upgrade to a Master!

Just browsing through the manual shows the similarity to White Knight, particularly that program's excellent command interface. There is no need to go to other menu screens to perform functions such as saving or reloading games, changing levels, sides or turning the board round. Even the colours used for the display, or the volume of the beep can be changed! It is extremely easy to set up positions for analysis or problem solving, and the current elapsed time for each player is shown at the right of the playing board. Sadly, I think the screen layout is inferior to that of the older



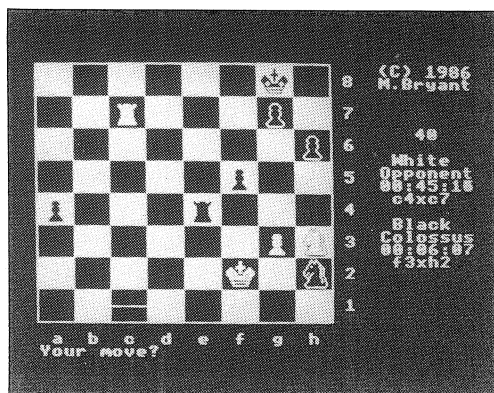
program: if you are playing white, your move and elapsed time is shown opposite the black pieces and vice-versa.

The manual indicates that the program version which loads in to a Model B or Electron is different to that for the Master or B+, and the board display shows this very clearly. The 32k program displays less than a full mode 4 screen as the program loads right up to £6040! On the 64k models mode 132 is used of course, which enables a 3-dimensional view of the board and pieces to be displayed. This is not the only difference, however, as a great many features of the larger program are omitted in the smaller. Even worse, some features present in White Knight are missing in 32k Colossus. In particular, the inability of the 32k program to save and reload games is a major drawback, and you should consider carefully if you are a Model B or Electron owner, as many features are unavailable.

Firstly, that 3D screen. After a few minutes, the novelty wears off, and with relief I used the 2D-screen option to revert to my familiar White Knight-style pieces. The problem is not with my monitor, but with the mode 132 graphics. Perhaps mode 128 would handle it better but then the text would be too small for TV sets. The other annoying thing about the 3D graphics is that it takes almost 10 seconds to draw the initial board, compared to under 1.5 seconds for the 2D board. So no bonus points for this fashionable feature.!

On the 64k version are the on-screen features which made White Knight such an

interesting program to play even when you are waiting for it to move. The line of play the computer considers best so far is shown (first move only with the 32k program), but even more interestingly, the large program shows what moves are currently being considered. This is quite an insight into how the program is performing its search, and quite a shock to see that this section of the display continues to flicker and change when you are thinking. It really is thinking in your time! The annoying thing, however, from a practical player's viewpoint, is that there seems to be no way to turn off this "hint". There is an option to vary the amount of information displayed here, and contrary to the implication of the manual I observe that the playing speed is 4-5 times slower when showing the full list.



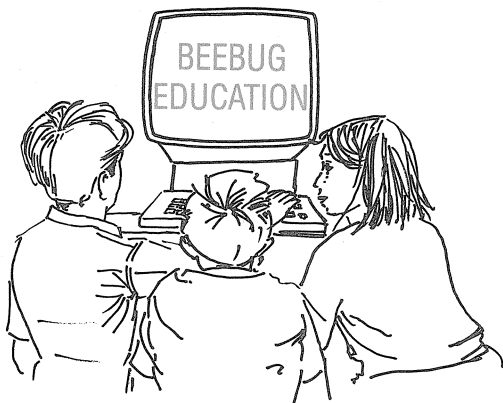
Since one of White Knight's weaknesses had been a lack of stored opening sequences, I was pleased to find that the 64k version has a small (3000 move) book of opening moves available. In practice, however, this is a little disappointing and does not seem to be very varied. For instance, for the first move it chooses in about equal proportion between e2e4, d2d4, and b1c3; with a rare try at c2c4. In reply to e2e4 (again in approximately equal proportion) we have e7e5; e7e6; g8f6; b8c6; d7d5; with a rare c7c5. While I respect Martin Bryant's quoted viewpoint that a careful choice of opening repertoire is necessary to give the best computer play, I feel that most players will want a more varied selection to practise against. At least pressing the U key will make the computer select another instead of the rare b1c3.

The White Knight owner would find himself familiar with the take-back (and forward) facility; changing of time-clock; legal move demonstration; play-self and supervisor modes, and invisible pieces (now expanded to leave one side still visible). New options are available to watch a replay of your defeat (?), and (large version) have the computer recant and choose a different move. More importantly, the modes-of-play options have been expanded: options are still available to set firm time limits (e.g. 40 moves in 2 hours); set average time limits (e.g. 1 move per 3 minutes); or match your play rate, but an 'infinite' mode has been added, and, more excitingly, a limited-time mode (e.g. all the moves in 10 minutes each). If you use the latter I advise you to set the computer's clock forward slightly to compensate for your move entry time - and beware, Colossus claims a win if you run out of time! Also new is the expansion of the mate-finding mode to include self-mates and help-mates, while disc versions also come with files of 34 famous computer games and 19 problems, but only 64k users can load them!

But what of the most important aspect - the standard of play? The manual claims a grading of 1850 ELO/156 BCF and that it beats White Knight Mk12 75% (Mk11 93%) of the time. It would need some weeks to evaluate the all-round strength play of a program but nothing I have seen leads me to disbelieve the above. The endgame algorithms have also been improved and I can confirm the manual's claim to be able to win the bishop+knight mate but on the 10-second move level it couldn't win the King+Pawn v King ending against me.

So is it a good buy? For the Master or B+ owner I must give a resounding "Yes" and endorse the title screen's claim to be the best chess program for home computers, and at a remarkably low price for such a major piece of software design. But for the Model B owner only a "Possibly". If you are prepared to play without a game-save option and don't want to watch the analysis, then the better play and limited-time modes will make your purchase worthwhile. And of course, it will still work (and better) if you do later upgrade to the Master or B+.

B



Mark Sealey, a teacher involved in many aspects of computers in schools, introduces the first of our bi-monthly reports on the education scene with an appraisal of the BBC's Domesday Project, and a look at the Grass Database from Newman College.

Introduction

Last month's magazine contained the first of our regular bi-monthly sections on communications. This issue sees the start of the other special area to which BEEBUG will be devoting more time every other month: BEEBUG Education will cover practical computing matters relevant to all areas of school and college use. We shall also deal with home study, distance learning and topics relevant, for example, to the Open University.

Ideas for future pages include Econet, the growing add-ons and extensions for use with LOGO, the use of Acorn family computers in Special Education, as well as some more interesting specialist developments in both the primary and secondary classroom.

BEEBUG Education will cater for the Master and Master Compact as well as the BBC B, B+ and Electron. If you are in any way involved yourself with children or students, in learning, teaching or education, then BEEBUG Education will have something for you. Indeed, please write to BEEBUG's editorial address and let us know of any topic of particular interest to you. News and developments in the educational computing field will be

featured as well as reviews of particularly interesting items of software.

This month sees an excellent database package called 'Grass' from Newman College under scrutiny. And we look at the much heralded AIV - Acorn and Philips' Advanced Interactive Video linked to last year's celebration of the 900th anniversary of the completion of the Domesday Book; some one million people in Britain took part in compiling the data that is now part of this twentieth century equivalent. Its scope is breathtaking; it has been estimated that a good reader working 18 hours a day would take seven years to read all the material contained on the discs. It is unique in the world in an area - interactive video - that many believe will be the next major development of computer use in education. And there is the Master controlling it all.

The Domesday Project

System : Acorn Master AIV micro,
BBC LaserVision Disc Player
14" Colour Monitor, Domesday
Disc Package, and User Guide

Supplier: Acorn Computers Ltd
Cambridge Techno Park,
645 Newmarket Road,
Cambridge CM5 8PD
Tel. (0223) 214411

Price : £3990 exc VAT (list price)
£2995 exc VAT (to education)
£1495 exc VAT (one per LEA)

The Master is in at the start not just of the Domesday Discs but many more that can be used with this same equipment: a BBC ecology game disc, for instance, is already being completed.

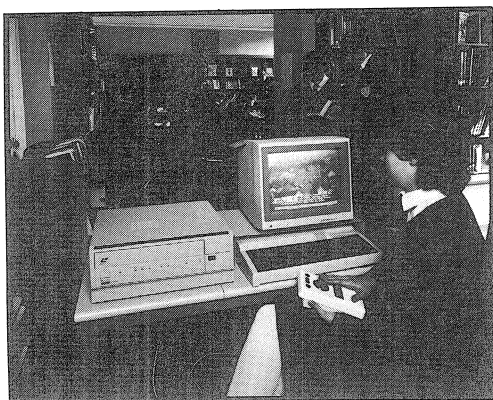
The Domesday project equipment under consideration here is the outcome of collaboration between Acorn, BBC Enterprises and Philips. Funding has come from the last two in addition to the Department of Trade and Industry. It would be nice - given the scope of what's on offer - to think that the impact will be similar to the original Micros in Schools scheme. Yet nobody really expects that it will.

THE EQUIPMENT

The chief reason is the price. Although there now exists an offer similar to the half-price 'Micros in Schools' scheme, this is limited to one per LEA only, and

the cost is still a minimum of nearly £2,000. This includes a Master with Turbo Co-Processor and an extra Small Computer Systems Interface (SCSI) which - together with the new Video Filing System ROM - allows interfacing to the BBC/Philips 415 LaserVision disc player. It is also possible to upgrade an existing Master to the Master AIV.

A Trackerball is used to select items for display. When buying the system complete, a Philips 14" medium resolution monitor is supplied. This CM8553 model allows more control of picture quality than is usual with monitors solely for computer use. This is because the monitor displays more than just digital computer output. The 'software' of the system -



containing up to 324 Megabytes of digital data as well as over 50,000 analogue video frames - is contained on two 12" LV discs. Each is double-sided so there is the small disadvantage of having to turn over the Community disc to change from North to South! Analogue images are combined physically with the familiar BBC character fonts to superimpose, say, captions on one of the 22,000 photographs. This is the achievement of this integrated system.

The content of these discs will disappoint no-one. For comprehensiveness it really is second to nothing of its kind in the World! Most of it has been specially commissioned by experts or is of government, public or (semi) official origin. Yet the depth and breadth of the subject areas that I looked at leaves

databases like Prestel and Knowledge Index standing.

The information is catalogued so that it is accessible with startling ease in a variety of ways; I found the trackerball easier to use in this context than a mouse, though the latter can be attached instead, and eventually a touchscreen. Data is downloaded at a rate of 150 Kbps. So the average time to explore any one geographical area or set of information really is only a few seconds.

In order to retrieve or browse, you can either zoom in from a large scale map or from a data index, and select from a menu displayed continually at the foot of the screen. Or you can search with alarming speed on a variety of criteria. Exemplary user-friendliness.

Not only are whole "journeys" based on a particular geographical area possible, but also the option of remodelling data from the huge array of statistics (cultural, population, economic, environmental and social etc) from the National disc. The Master can then overlay these back onto a selected map or town. I understand that the BBC plans to bring out software that will allow the user to download this data onto a standard disc.

WHO WILL USE IT?

The system will obviously appeal to the one million people (including the pupils in 14,000 schools) who helped to gather the data two years ago. It will appeal to anyone interested in laser technology and interactive video; libraries will want it, tourist offices and agencies, not to mention commerce, land and estate agents and industry. It is for anyone concerned to have a reliable comprehensive electronic encyclopaedia that is extremely easy and exciting to use.

CONCLUSION

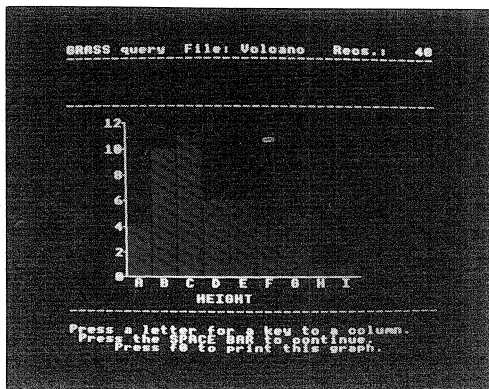
My major misgiving has to remain the inaccessibility. If a resource like this - appropriate to so many sectors of education and beyond - is centralised in the larger colleges and university departments and computer centres (which its cost may well determine), many would-be users of this excellent resource will be left in frustration.

The Grass Database

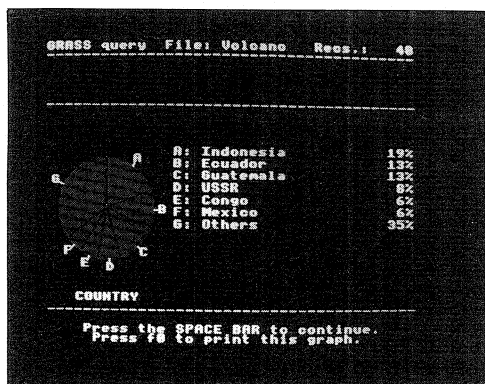
Product: Grass Database
Supplier: Newman College
 Bartley Green,
 Birmingham B32 3NT.
 Tel. 021-476 1181
Price: £25.00 plus VAT
 (bulk licences available)

For Primary and many Secondary school teachers news of the software reviewed this month will be very welcome. Newman College's Information handling package Grass (Graphics Searching and Sorting) fills a comparatively wide yet precise gap in the market.

Grass is essentially a database package written in Basic that has all the usual data entry, retrieval, sort and print facilities, yet one that is extremely easy to use. Selection of options is of the position-cursor and space-bar-to-select kind. Running on Acorn DFS version 0.9 as well as 1.2, it will also work with Econet Level 2 with versions 3.34 or 3.6 of the Network Filing System. It is happier without the second processor, but files can be converted to and from Quest format.



The distribution disc comes with five sample files and the instructions for creating and editing your own are very clear and easy to follow. There are inevitable limitations: it is not possible to use a separate DATA disc as such, and consequently with 40 track single-sided discs, space will be an important factor. The maximum number of fields is 18. The



maximum number of records is 256 - just 100 if all 18 fields are used. It is impossible to add a field at a later date, as you may using ViewStore, say.

From this it ought to be obvious that for advanced or sophisticated work, Grass is not, perhaps, the best possible package. I have used it successfully with 11 year olds; as an introduction to information handling on the computer it has much to recommend it. Top of the list must be ease and reliability of use. Speed in operations is also a plus.

Where it really scores, though, is on the graphics side. Once data is entered as desired, it can be displayed visually in a variety of ways (see illustrations). If, for example, you had a database on aeroplanes as a follow-up to some visit or part of a project, you could display wing-spans, ages or maximum speeds in a whole host of formats: histogram, cumulative, count or scatter graph, pie charts and the like. Averages and medians can also be dealt with. In selecting and using these displays, Grass will prompt with messages both as to the types allowed for any one field and when the data is either too little or too much for meaningful results. These can then be dumped to a printer - including a Network printer. Parameters are set up via a "Teacher's Page"!

For the teacher who wants to introduce his/her pupils to the basic use of a computer in manipulating data safely and with immediate powers of interpretation through graphic display Grass is very strongly recommended.

D
D

MULTI-CHARACTER PRINTER DRIVER FOR

Claus Alsted has devised a printer driver for View that not only copes with all the usual fonts, but allows graphics and foreign language characters to be printed too.

View has always been one of the two most popular word processors for the BBC micro, along with Wordwise (and Wordwise Plus). As View is now supplied by Acorn as standard with the Master 128 and Compact computers, the number of View users looks set to grow even further.

One of the advantages (sometimes a disadvantage) of View is that any text is prepared largely independent of the printer used to produce hard copy output. Provided only that a simple text layout/style is required, there are no problems, but if you wish to make full use of your printer's capabilities (different sizes and styles of text, graphics characters etc) then a View 'printer driver' is required. This translates your printing specifications into the correct control codes required by your printer. Acorn supply (at a price) a printer driver package that can be tailored to most printers, and BEEBUG published a useful printer driver generator in Vol.4 No.3.

Even so, conventional printer drivers seldom give access to all of a printer's facilities, and have other limitations. This is mostly because all the required code sequences have to be stored as part of the printer driver itself.

The printer driver program listed here is quite different. It is itself largely independent of which printer it is used with, and gives access to virtually all the facilities that your printer can provide. These include not only the normal text features like condensed, underlined, double height, etc, but also the use of graphics characters (if provided by your printer) and foreign language character sets as well.

INSTALLING THE PRINTER DRIVER

Type in the program as listed (the spaces between the line number and the following assembly language instruction may be omitted for convenience) and save to cassette or disc for security. Now run the program to assemble the printer driver and follow the screen instructions to save the printer driver with the name of your choice. From within View, the printer driver can be loaded at any time with the PRINTER command (see the various View manuals for more information).

The printer driver program was written for a Shinwa CP80 printer, compatible with the Epson MX series of printers. All Escape sequences use ASCII 155 (27 + 128) for the ESC character and this is set in the program at line 1530. If your printer uses ASCII 27 in Escape sequences then replace 155 by 27 at this point, and wherever else it occurs.

The only other printer-specific settings are shown in the following table together with the line number in which they are set. If your printer uses a different code, then change the value in the program to that used by your printer.

A comprehensive demonstration (with examples of the use of highlight characters) is contained on page 20. This is described later in the text.

```

10 REM Program DRIVER
20 REM Version B1.4
30 REM Author Claus Alsted
40 REM BEEBUG March 1987
50 REM Program subject to copyright
60 :
100 ON ERROR ON ERROR OFF:IF ERR<>17 R
EPORT:PRINT" at line ";ERL ELSE PRINT':E
ND
110 code = &3F00:REM assembly location
120 PROCassemble
130 PRINT"End of code: "&";~drvfinito;
140 IF drvfinito<&500 PRINT" OK " ELS
E PRINT" Too long!":END
150 INPUT"Printer name: "P$
160 save$="SAVE "+P$+" "+STR$~code+" "
+STR$~(code+255)+CHR$32+"400"
170 PRINT"Ready to *save$" (Y/N) "
180 REPEAT:G=GET:UNTIL G=89 OR G=78:IF
G=89 OSCLI save$
190 END
200 :

```

ESC O 1260 Skip-over-perf
cancel
ESC E/F 1890 Emphasize
on/off
ESC - 1960 Underscore
on/off
ESC @ 2050 Printer init-
ialisation

USING THE PRINTER DRIVER

View allows the use of two 'highlight' markers. These are shown on the screen as '*' and '-' (inverted in later versions of View), and produced using Shift-f4 and Shift-f5 in edit mode. In addition, the meaning of either of these two highlights can be changed using the embedded command 'HT'. The default highlights are underscored (-) and emphasized (*) text. With the printer driver listed, the two highlight characters have the following meanings.

HT h 128 underscored text
HT h 129 emphasized text
HT h 130 ASCII multi-toggle
HT h 131 ASCII solo-toggle
HT h 155 Escape sequence
(or HT h 27)

'h' denotes either of the two highlight characters '-' or '*'. An "HT - 155" command allows all Escape sequences, for example, to be given as embedded commands prefixed with '-' (Shift-f4). "HT * 130" or "HT * 131" allows '*' (Shift-f5) to act as a toggle, either on the next character only (single toggle) or on all the following characters until reset with '*' (multi-toggle). Of course, either highlight, '-' or '*', can be defined as required. The toggle control provides a means of generating ASCII codes that cannot otherwise be entered directly from the keyboard:

```

1000 DEFPROCassemble
1010 REM System addresses:
1020 osasci=&FFE3:oswrch=&FFEE:osbyte=&FFF4
1030 FOR pass=4 TO 7 STEP 3
1040 O%=code:P%=&400:REM assemble at code, RUN at &400
1050 OPT pass
1060 .start \jump table 13 B
1070 JMP print
1080 .prton
1090 JMP on
1100 .prtoff
1110 JMP off
1120 .hmi
1130 NOP:NOP:NOP
1140 .option
1150 RTS
1160 .on \ 29 B
1170 LDA #0 \reset flags & toggles
1180 STA multiflg + 1
1190 STA singleflg + 1
1200 STA uscore + 1
1210 LDA #ASC"F"
1220 STA bold + 1 \all toggles reset
1230 LDA #2 \printer ON
1240 JSR oswrch
1250 JSR prtreset \reset printer
1260 LDA ASC"O" \zero skip over perforation
1270 JMP prtesc \ESC "O" to printer
1280 .off \ 8 B
1290 JSR prtreset \reset printer
1300 LDA #3 \printer OFF
1310 JMP oswrch
1320 .print \ 77 B
1330 PHA \save registers
1340 STA acc + 1
1350 TXA: PHA
1360 TYA: PHA
1370 .acc
1380 LDA #0 \dummy value
1390 BPL nthilite \A < 128
1400 CMP #129
1410 .ht128 \HT-
1420 BCC underscore \A = 128
1430 .ht129 \HT*
1440 BEQ emphasize \A = 129
1450 CMP #131
1460 .ht130 \HT * 130'
1470 BCC multiflg \A = 130
1480 BNE ht155
1490 .ht131 \HT * 131'
1500 LDX #1 \A = 131
1510 STX singleflg + 1 \set singleflg ON
1520 .ht155 \HT - 155'
1530 CMP #155
1540 BEQ printit \A = 155, ESC on CP80
1550 BNE restore \ignore everything else
1560 .nthilite
1570 LDX multiflg + 1
1580 BNE toggle \toggle if multiflg <> 0 (= 1)
1590 .singleflg

```

```

1600 LDX #0 \singleflg initially OFF
1610 BEQ printit \no toggle if singleflg=0 (OFF)
1620 DEX \singleflg was ON - now reset
1630 STX singleflg + 1 \singleflg and
1640 STX multiflg + 1 \multiflg
1650 .toggle
1660 CMP #64
1670 BCC printit \A < 64 - print as is
1680 CMP #96
1690 AND #&1F \mask lower 5 bits (0 - 31)
1700 BCS loctrl \A>=96 - print as low Ctrl char
1710 ORA #&80 \set MSB - hi Ctrl char
1720 .printit
1730 JSR osascii \print as ordinary char
1740 .restore
1750 PLA: TAY
1760 PLA: TAX
1770 PLA
1780 RTS
1790 .loctrl
1800 JSR prtonly \low Ctrl chars to printer only
1810 BPL restore \JMPrel
1820 .multiflg \command HT * 130 9 B
1830 LDA #0 \initially OFF
1840 EOR #1 \toggle 0/1
1850 STA multiflg + 1 \save new value
1860 BPL restore \JMPrel
1870 .emphasize \HT* 12 B
1880 .bold
1890 LDA #ASC"F" \initially OFF
1900 EOR #3 \toggle "E"/"F"
1910 STA bold + 1 \save new value
1920 .emph
1930 JSR prtesc \print ESC, ASC(Acc)
1940 BPL restore \JMPrel
1950 .underscore \HT- 17 B
1960 LDA #ASC"- "
1970 JSR prtesc \ print ESC "- "
1980 .uscore
1990 LDA #0 \initially OFF
2000 EOR #1 \toggle 1/0
2010 STA uscore + 1 \save new value
2020 JSR prtonly
2030 BPL restore \JMPrel
2040 .prtreset \reset printer 19 B
2050 LDA #ASC"@ "
2060 .prtesc \ send ESC & CHR$(Acc) to
2070 PHA
2080 LDA #27
2090 JSR prtonly
2100 PLA
2110 .prtonly \printer only
2120 PHA
2130 LDA #1
2140 JSR oswrch
2150 PLA
2160 JMP oswrch
2170 .drvfinito
2180 ]
2190 NEXT:ENDPROC

```

ASCII values	toggled into
32 - 63	32 - 63
64 - 95	128 - 159
	(high control)
96 - 127	0 - 31
	(low control)

For example, enlarged or wide characters are produced by the Epson FX80 printer with ESC W followed by ASCII 1 or 48. This can be specified with -W*a ('a' has an ASCII value of 97 which is toggled by '*' to 'l'), and switched off with -W* (which is ESC W 0). Further examples of the use of the extended highlights are shown in the accompanying illustration. Note that '-' and '*' indicate the use of Shift-f4 and Shift-f5 except for 'underscore on' which is Shift-f4 followed by '_'. A View file containing notes on the extended highlights, the examples and a keystrip for the CP80 printer is included on the magazine cassette/disc along with the printer driver program. The keystrip can be easily edited to suit your own printer.

Although the technique may sound complex, it soon becomes relatively easy in practice, though at the cost of losing a correctly formatted text screen, usually one of the big pluses for View. You will, of course, need to check the manual for your own printer carefully, as the effects produced are determined by that device and not by the printer driver program. With care, and by experimenting, you should now be able to control all the features of your printer from within View.

The printout reproduced full size below was obtained using the Printer Driver with a Shinwa CP80 printer, and shows graphics characters and foreign language character sets.

Graphics characters

```

*~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
*~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~ *~
  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z  [  \  ]  ^  _  `  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z  {  |  }  ~

```

Fonts

	35	36	64	91	92	93	94	96	123	124	125	126
-R*Q US ASCII:	#	\$	@	(\]	^	'	(!	}	~
-R*A French :	#	\$	à	°	ç	š	^	'	é	ù	é	~
-R*B German :	#	\$	ä	ä	ö	ü	^	'	ä	ö	ü	ß
-R*C English :	#	\$	£	£	£	£	^	'	£	£	£	~
-R*D Danish :	#	\$	ø	ø	ø	ø	^	'	ø	ø	ø	~
-R*E Swedish :	#	\$	é	é	é	é	^	'	é	é	é	ü
-R*F Italian :	#	\$	é	é	é	é	^	'	é	é	é	ü
-R*G Spanish :	#	\$	@	!	ñ	¿	^	'	ü	ü	ü	~

ASCII codes

The printout reproduced full size below shows a variety of fonts and sizes obtained with the printer driver and an Epson FX80 printer.

ON	ESC Function	OFF
---*A	UNDERSCORE	---*@
-E	EMPHASIZED	-F
-G	DOUBLE STRIKE	-H
-W*~	WIDE	-W*'
-S*Q	SUPERSCRIPT	-T
-S*A	SUBSCRIPT	-T

ON	CTRL Function	OFF
*N	WIDE	*T
*O	CONDENSED	*R
*G	BELL	

		NORMAL PITCH
		NORMAL PITCH

New OS Calls Uncovered

Following last month's list of the Beeb's OS vectors, Claus Alsted reveals some useful but undocumented OS routines in the Model B, and Roger Cullis supplies the equivalents for the B+, Master and Compact.

Many people will be familiar with the "official" OS machine code routines and their entry points - OSWRCH, OSBYTE, OSCLI, OSRDCH etc. But the operating system has other routines which are just as easy to utilise. The only drawback is that their addresses change with each new release of the operating system. But in many cases this disadvantage is offset by the sheer convenience of the routines themselves.

In the accompanying table we present three groups of routines, accessed through six possible entry points; and these are given for the four most common machine

configurations. Each entry point has been christened with a suggested label for reference purposes. And at the end of the article we give one example of the extreme economy offered by the routines in writing a debugging aid.

DISPLAY IN HEX FORMAT

This useful group of entry points is concerned with outputting hex values in ASCII form. We will start with OSWRHX (OS Write Hex) which prints the contents of the accumulator in hex notation. Thus the code:

```
LDA #&FE: JSR OSWRHX
```

will print "FE" on the screen. Owners of a model B or B+ may also use OSWRHXS. This has the same effect as OSWRHX except that a leading space is also inserted. Thus:

```
LDA #&FE: JSR OSWRHXS
```

will print " FE" on a model B or B+. For the sake of comparison, we should mention that OSWRCH writes the contents of the accumulator directly to the screen as if it contained an ASCII code. Thus, calling it with &FE in the accumulator would send code 254 to the screen.

LABEL	B	B+	MAST	COMP	FUNCTION	CONDITIONS
OSWRHXS	&F975	&F97E	none	none	Print a space then...	X and Y preserved
OSWRHX	&F97A	&F983	&BD6C	&BCCC	Print the contents of Accumulator in hex	A undefined
OSWRHXN	&F983	&F98C	&BD75	&BCD5	Print contents of LS nibble of Acc in hex	ditto
OSWRMS	&FA4A	&FA48	&BECF	&BE2F	Print the following embedded message of any length, terminated by zero	Y<>0 on entry Y preserved A, X undefined
OSACK	&F9A6	&F9AE	&B285	&B1DB	Acknowledge Escape (*FX126) then...	none
OSESC	&F9AB	&F9B3	&9B7D	&9B5D	ESC error message - error code 17	none

The third entry point of this group, which we have called OSWRHXX (OS Write Hex of least significant Nibble), will print in hexadecimal notation the value of the least significant nibble of the accumulator. Thus:

```
LDA #&FE:JSR OSWRHXX
will print "E" on the screen.
```

WRITE ASCII MESSAGE

The routine which we have called OSWRMS (OS Write MeSsage) has a similar effect to the PRINT statement in Basic. It will handle messages of any length, which may even contain carriage returns. It will in fact output any byte value except zero, which is used as a terminator. If you wish to send a VDU sequence which contains a zero byte, you can sometimes replace the zero with 2^n. For example VDU22,8 has the same effect as VDU22,0. Note that the message should immediately follow the JSR to OSWRMS, and that it should be called with a non-zero value in the Y register:

```
LDY #&FF \ Y <> 0
JSR OSWRMS
EQUUS "This is the message ..."
EQUB 0
... \ rest of program
```

The effect of the above will be to print "This is the message ..." and then to continue the program.

HANDLING ESCAPE

The third group of calls (OSACK and OSESC) handles Escape conditions from machine code. The most obvious use for this is after reading keyboard input with OSRDCH:

```
JSR OSRDCH \ read input
CMP #27 \ was it Escape?
BNE ntesc \ no - skip
JMP OSACK \ yes - acknowledge and
           print error message
           then quit
.ntesc
... \ rest of program
```

APPLICATION: A REGISTER DISPLAY

When debugging machine code programs it is often essential to check the contents of the major registers from time to time. If you have no machine code toolkit, the following simple routine will probably be very handy. It prints out the contents of the P, A, X and Y registers in hex, making extensive use of OSWRHXX.

The register display routine is listed from line 200 to 260. To show how it

works, it is accompanied by a short piece of code to test it out; and you will notice that INKEY(-256) is used to determine which computer system you are using, and to correct the address of OSWRHXX accordingly.

When using the routine to debug your own code, include lines 210 to 260 somewhere within your machine code, having already supplied the correct address of OSWRHXX. Then, each time that you need to know the contents of the Accumulator and the X, Y and Status registers, just include the following statement:

```
JSR help
```

The result is printed in hex in the order P, A, X, Y; and the routine preserves all registers intact. Incidentally, model B and B+ users can make the routine even more concise if they wish by inserting the line:

```
write=OSWRHXS
```

early in the program, after first defining OSWRHXS, and deleting lines 250 and 260.

We hope in future issues of the magazine to include further OS routine entry points.

```
10 REM P, A, X & Y Register Display
20 REM Using OSWRHXX ver 7
30 MODE7
40 OSNEWL=&FFE7:OSASCI=&PFE3
50 DIM R% 100
60 A=INKEY(-256)
70 IF A=-1 OSWRHXX=&F97A:REM B
80 IF A=251 OSWRHXX=&F983:REM B+
90 IF A=253 OSWRHXX=&BD6C:REM MASTER
100 IF A=245 OSWRHXX=&BCCC:REM COMPACT
110 FOR I=0 TO 3 STEP 3
120 P%=R%[:OPT I
130 .entry
140 LDX #0:LDY #&FF:LDA #&AA
150 JSR help
160 LDX #&FF:LDY #0:LDA #&11
170 JSR help
180 RTS
190 :
200 \ Register Display Routine
210 .help:PHP:PHA:PHA:PHP:PLA
220 JSR write:PLA:JSR write:TXA
230 JSR write:TYA:JSR write
240 JSR OSNEWL:PLA:PLP:RTS
250 .write:JSR OSWRHXX:LDA #&20
260 JSR OSASCI:RTS
270 :
280 ]:NEXT
290 CALL entry
```

The ADE+ Assembler System

ADE+ is a complete system for the machine code programmer. Geoff Bains explains what the package provides, and gives his own assessment.

Product: ADE+

Supplier: System Software
12 Collegiate Crescent,
Sheffield S10 2BA
(0742) 682321

Prices : £49.55 – Disc (SWR image)
£54.15 – ROM
£57.60 – Master cartridge

System's ADE+ is more than just a machine code assembler. It is a complete piece of system software in its own right.

ADE+ consists of 32K of ROM image (in one form or another, depending on your choice of format) and a utility disc. The program is in four parts - a 'memory management unit, text editor, assembler, and a debugger. The debugger is System's old SPY package offering basic machine code monitor facilities.

Of the rest of the software the overall control is with the memory management unit (MMU). It is this that is called with the inevitable *ADE command.

The MMU sorts out the memory used by ADE+. The MMU also runs a continuous real time clock (using the hardware clock on a Master) which appears as the prompt for the MMU command level. Some of the commands available are simply to call other sections of the system - EDIT calls the editor, DEBUG summons the debugger, and ASM sets the assembler going. Other MMU commands deal with setting up the micro for best use with ADE+.

All the memory available is split into separate sections - workspace, and input, output and printer buffers. This memory

can come from main RAM, sideways RAM and second processor RAM. If shadow RAM (either Acorn or Aries) is available (and enabled) then this is taken into account.

The workspace is usually the main part of the available memory and is used (reasonably enough) for all the workings of the various ADE+ sections. The input buffer is used for all data entering the ADE+ system. This includes, for example, an assembler program coming in from disc for assembly. The output buffer would in this case form the buffer for the assembled code to accumulate before output to a disc file.

The printer buffer is made up of all available sideways RAM that has not been protected (because it, say, contains a sideways ROM image). ADE+ contains a print spooler that operates as a background task allowing much faster use of the system when printing. This uses the printer buffer for temporary store.

The buffers default to zero size on a single processor system. However, they can be given sizes by the user with MMU commands or they will each grab 1K of RAM from the workspace when needed if they are not allocated a specific size. The larger a buffer is, the quicker connected operations will be.

The next section of the ADE+ system a user will meet is the editor. ADE+ can use any editor that produces ASCII output without embedded codes. View is especially catered for as the system can recognise its rulers. View can also be recognised for material to be passed automatically to it for editing. However, details on how to do this are only contained in the 'Advanced' ADE+ guide.

If you have no favourite editor one is provided in the ADE+ ROMs. System says this is a 'simple' editor. However, it is quite complex enough for most purposes. The editor operates in the normal way. It has a command mode for loading and saving text and other operating commands (such as calling the assembler) and an editing mode. In the editing mode the cursor keys allow you to move around the text and insert or overwrite any corrections or additions.

A block of text can be defined with markers and then moved, copied or deleted with the function keys. Individual lines or characters can be deleted or blank ones inserted. A search facility is also provided for moving the cursor to occurrences of a specified string.

The editor provides one function specifically for editing an assembler program: an assembler label can be searched for. Pressing f2 with the cursor on a line with a label reference moves the cursor to the line containing the label itself. This is an extremely useful feature and saves much frustrating searching through a program, especially if it's an old one, or someone else's.

ADE+ accepts assembler code in the usual format of lines made up of four fields - label, opcode, operand and then comment. Each field must be separated with one or more spaces and/or a tab character. Tabs are expanded to positions every eight characters although this can be altered with a View format ruler. The whole operation of the editor is very much like View itself, and it is obvious that Acorn's program provided System with the model.

Existing BBC Basic assembler programs can also be loaded into the editor after conversion with a utility provided on disc. Once a program has been edited it is assembled. This can be done either from within the editor's command mode or from the MMU. Either way, the ADE+ assembler is extremely fast. System claims ADE+ is faster than all rival products. This certainly seems likely.

It is the assembler which really shows off ADE+'s power. ADE+ is not just a simple assembler like the one built into BBC Basic. It is more of a compiler than an assembler. The pseudo ops provided and the modular nature make using the ADE+ assembler more like using a high level language than a simple assembler.

ADE+ has over 60 pseudo ops available. As well as the likes of ORG to define the start address of the object code and EQU to insert an item of data, ADE+ includes unexpected codes like QUERY to input data

from the keyboard at assembly and even WHILE...WEND for repeating assembler sections.

ADE+ assembler programs can be written in one of three ways. As with any assembler a program can be written, assembled and then run. The problem with this method is that the program length is limited by the RAM available. Other systems get around this problem with 'macros' - self-contained subroutines of assembler code that can be defined and then called up by name from another assembler program. ADE+ supports macros but also goes a stage further.

ADE+ can create and use 'link modules'. These are not fully assembled when put through the ADE+ assembler. Instead it just works how much room they will take up and what variables and labels they will use. This information can then be saved and many link modules assembled together with the special linker program in the ADE+ ROMs.

Because subroutines 'assembled' in this way are not yet fixed as to address, variable values, label locations and so on, they can be used as genuine relocatable library routines. Once a routine has been developed and tested in this way it can be kept for use at a later date in any other program, confident that it will work.

Not only is ADE+ compatible with all versions of the BBC micro but it can make the most of each too. It can use all the 'extended' codes used by the second processors 65C02 and the Masters' 65C12, and the extra codes provided on only the Rockwell microprocessors. All this makes the ADE+ system the most comprehensive around. However, it is not a program for the faint-hearted. The ADE+ manual is heavy work to plough through. It is not well explained and poorly structured.

Nevertheless, I have little hesitation in recommending this package. Until ADE+, Clares' Macrom was the last word in assemblers for the Beeb. It is clear this program provides a complete development system for a remarkably low price.

DD

VIRTUAL ARRAYS (Part 2)

Jacob Abboud describes some alternative but more sophisticated ideas to conclude his discussion of virtual array techniques for data storage.

In the first part of this article, we looked at the 'separate files' method for implementing virtual storage of large arrays. In this final part, we look at two alternative and more elegant methods.

STACKED RANDOM ACCESS FILES

The 'separate files' method, described last month, stores each array in a separate data file, and is thus limited to a maximum of five arrays. The 'stacked random-access files' method, however, stacks all the arrays in a single file starting with the array with reference "A". The stacking sequence is very important for setting the pointer when accessing the arrays, and it is essential to use the correct array reference.

In order to implement this method, programs (1) and (2) from last month should be modified as shown opposite. Make these alterations and run the driver using the same example given in part one, with two integer or real arrays A(100), B(100,3). The performance of this method is similar to that described in part one (in fact, one second slower when writing the 400 numbers due to the increased calculations introduced by PROCpointer). This method provides the flexibility of using only one file to store a number of arrays, and is thus limited only by the available disc space.

PAGED VIRTUAL MEMORY

This is the most complex of the three methods for implementing virtual storage. It departs completely from the unblocked 'random-access files' techniques applied in the previous two methods. We therefore need a new virtual memory driver and main program procedures.

THE VIRTUAL MEMORY DRIVER PROGRAM 1

```
Insert 1045 X=OPENOUT("A")
Delete lines 1080,1160
Change 1170 BPUT#Z,s1%,BPUT#Z,s2%,
      BPUT#Z,it%
Insert 1195 CLOSE#X
```

DEMONSTRATION PROGRAM 2

The main function of PROCopen now changes to reading the information from !info and installing it in &70-&9A (assuming that the number of arrays is less than 10), and to opening the one file containing the stacked arrays, file "A". It is also necessary to include a new procedure to calculate the position of the pointer (PROCpointer), and hence changes to both PROCout and FNin. The parameters for accessing the arrays remain unchanged. The full listings of PROCopen, PROCout, FNin, and PROCpointer are included below to avoid confusion:

```
1000 DEF PROCopen
1010 Z=OPENUP("!info"):X=OPENUP("A")
1020 n%=BGET#Z
1030 FOR I%=1 TO n%
1060 I%?&6F=BGET#Z:I%?&7F=BGET#Z
1070 I%?&8F=BGET#Z
1080 IF I%?&8F=0 I%?&8F=5 ELSE I%?&8F=6
1090 NEXT:PRINT" Ready"
1100 ENDPROC
1110 :
1120 DEF PROCout(i%,j%,b,A$)
1130 a%=ASC(A$)-64:PROCpointer
1140 IF a%?&8F=6 PRINT#X,b
1150 IF a%?&8F=5 b%=b:PRINT#X,b%
1160 ENDPROC
1170 :
1180 DEF FNin(i%,j%,A$)
1190 a%=ASC(A$)-64:PROCpointer
1200 IF a%?&8F=6 INPUT#X,v:=v
1210 IF a%?&8F=5 INPUT#X,v%:=v%
1220 :
1230 DEF PROCpointer
1240 sz%=0:IF a%=1 GOTO 1280
1250 FOR K%=1 TO a%-1
1260 sz%=sz%+K%?&6F*K%?&7F*K%?&8F
1270 NEXT
1280 sz%=sz%+a%?&7F*a%?&8F*(i%-1)+a%?&8F
      *(j%-1)
1290 PTR#X=sz%:ENDPROC
```

EXPLANATION OF THE METHOD

The basic idea behind this scheme is to use a fast disc drive (preferably a RAM disc) as a primary storage device with

main memory acting as buffer space. Storage is divided into blocks, or pages, which are resident on disc. During computation, pages are brought into memory buffer space (which contains several blocks, not necessarily sequential, in the form of arrays), and eventually written back. If several blocks are resident in memory, and an element of an array is requested which is not present, the program chooses one of the current blocks for exchange with the appropriate one on disc. A poor choice of algorithm can result in continuous swapping of blocks (called thrashing). The present algorithm monitors the amount of access to each block and chooses the least used block for swapping. When a new block is brought into memory, it will, at that time, have had the least usage of all, and will, artificially, be most vulnerable to swapping. To avoid this, the new block is arbitrarily assigned a number of accesses equal to the current maximum, thereby ensuring its residence in memory for some time.

THE NEW DRIVER PROGRAM

Type in this month's driver program (Driver Program 3), and save it. The program does not require any inputs since data is built into the program (array dd%). The choice of the number of memory pages (pgm%) and page length (pgl%) will depend on the amount of memory available after writing your own applications program, and the number and size of the different arrays to be used.

Each array occupies pgm% pages in memory. The variable na% should be set to the number of arrays that will use virtual memory (in this case two), and the data at line 120 should show for each such array in turn, the second dimension (1 if necessary) and type (integer 0, real 1). For each array, dd%(i%,1) holds the second dimension of array i, and dd%(i%,2) the type. The driver program will generate a number of random access disc files equal to npd%*na% which will be named 11, 12, 21, 22, etc. In each case the first digit references the array, and the second the page. Hence, to accommodate X(500), we can have pgl%=50, and npd%=10, and so on. Make sure that the commands in lines 190 and 200 are correct for your filing system (RAM disc, DFS etc). Running the program should then be straightforward.

MAIN PROGRAM

Type in the second (demo) program and save it. This establishes two virtual arrays, called here X and Y. The procedures that you would need to include in your own program are PROCinit, PROCout, FNin, PROClimit, PROCpagin, PROCpagout, FNmax, FNmin. The rest of the program is a demonstration of the technique.

The working buffer in the main program should be implemented by using two or more arrays, dimensioned as X(pgm%,pgl%) if a single array, and Y(pgm%,pgl%,second dimension) if a two dimensional array (you can choose any array names you like, but in calling FNin and PROCout the first array is referenced as array '1', the second as array '2' and so on - the parameter a%). Choose as large a value of pgl% as you can get away with because it will improve the efficiency of the program. You can experiment by changing the values of pgm% and pgl% for optimum efficiency for your particular application. The program works by checking whether the requested element of an array is in its buffer, and if so in what page. If it is not present, it calculates the page required, and determines which page in buffer should be swapped with the required page on disc. If the page to be swapped has been written to, it is saved before the new page is loaded from disc.

For optimum efficiency, the routines would have to be rewritten in machine code, and possibly placed in sideways RAM. The demonstration program writes 2400 bytes (400 numbers) to two arrays and then reads them. During execution the pages in memory (1-3 for each array, pgm%=3, pgl%=25) are displayed so that you can see how the swapping takes place. The routine performed quite reasonably using a RAM disc. I found great improvement when I compiled the program using Computer Concepts' Accelerator.

ADAPTING THE PROGRAMS

To assist in adapting the programs to your own requirements we will look at a further example. Suppose we wish to use three virtual arrays called DAY, MONTH and YEAR, where MONTH is two dimensional and the other two are one dimensional. Set na% (in both programs) to 3. At line 120 in the Demo program dimension the three arrays DAY, MONTH and YEAR as described

before. Modify PROCout and FNin to accommodate the three arrays (lines 1130, 1140 and 1230, 1240):

```
IF a%=1 DAY(b%,pos%)=b
IF a%=2 MONTH(b%,pos%,j%)=b
IF a%=3 YEAR(b%,pos%)=b
```

PROCEDURES AND FUNCTIONS

init Initialises memory pages, and sets the write status to zero.

out Writes data to array. Calls PROCpagin if the element is not in the memory buffer, and finally increments number of accesses to array by 1.

in Reads data from array. Calls PROCpagin as in 'out'.

limit Works out whether the element is in one of the buffer pages and also finds its position in the buffer. If not, calculates the required page to be loaded.

pagin Calls PROCpagout to save page to be swapped if previously written to, before loading the required page from disc.

pagout Writes page to disc.

min Finds the least used page in buffer.

max Finds the maximum accesses of an array in buffer.

loc Finds location of data to be read from &70-&90.

VARIABLES

pgl% Page length in memory

pgm% Number of pages in memory

npd% Number of pages on disc

na% Number of arrays

acc%() Number of accesses array

i% First array dimension index

j% Second array dimension index

b,b% Number to be written by PROCout

v,v% Number returned by FNin

A\$ Array reference letter

a% Array reference number

w% Used as a write flag

b% Current page in buffer

lr% Lower limit of array in a page

hr% Higher limit

pgr% Page required

pos% Position of element in array

pg\$ Name of page on disc

mn% Minimum value of accesses

mx% Maximum value of accesses

L%,K%, Local counters

r%

DRIVER PROGRAM 3

```
10 REM VIRTUAL MEMORY DRIVER 3
20 REM Version B4.2
30 REM Author Jacob Abboud
40 REM BEEBUG March 1987
50 REM Program subject to copyright
60 :
100 pgl%=25:npd%=4:na%=2
110 DIM dd%(na%,2)
120 DATA 1,0,3,0
130 FOR I%=1 TO 2
140 FOR J%=1 TO na%
150 READ dd%(J%,I%)
160 NEXT
170 NEXT
180 :
190 *RAMDFS 1
200 *DR. 1
210 MODE7
220 FOR I%=1 TO npd%
230 FOR J%=1 TO na%
240 nm$=STR$(I%)+STR$(J%)
250 PRINTTAB(0,3)"Writing page ";nm$
260 z=OPENOUT(nm$)
270 FOR K%=1 TO pgl%
280 FOR L%=1 TO dd%(I,J%)
290 IF dd%(2,J%)=1 PRINT#z,le-20 ELSE
PRINT#z,0
300 NEXT
310 NEXT
320 CLOSE#z
330 NEXT
340 NEXT
350 PRINT'"Paged Virtual Memory Instal
led"
360 END
```

DEMO PROGRAM 3

```
10 REM PAGED VIRTUAL MEMORY DEMO 3
20 REM Version B1.3
30 REM Author Jacob Abboud
40 REM BEEBUG March 1987
50 REM Program subject to copyright
60 :
100 MODE7
110 @%=4:pgl%=25:pgm%=3:na%=2
120 DIM X(pgm%,pgl%),Y(pgm%,pgl%,3)
130 DIM acc%(pgm%,na%)
140 PROCinit:TIME=0
150 FOR I%=1 TO 100
160 PROCOut(I%,1,I%,1)
170 NEXT
180 FOR I%=1 TO 100
190 FOR J%=1 TO 3
200 PROCOut(I%,J%,I%,2)
210 NEXT
220 NEXT
```

```

230 t1=(TIME DIV 100)MOD 60:TIME=0
240 :
250 I%=0
260 REPEAT
270 I%=I%+1
280 X=FNin(I%,1,1)
290 UNTIL I%=100
300 :
310 I%=0:J%=0
320 REPEAT
330 I%=I%+1:J%=J%+1
340 Y=FNin(I%,J%,2):IF J%=3 J%=1
350 UNTIL I%=100
360 :
370 t2=(TIME DIV 100)MOD 60
380 CLOSE#0
390 PRINT"CPU time taken:"
400 PRINT"Writing data ";t1
410 PRINT"Reading data ";t2
420 END
430 :
1000 DEF PROCinit
1010 FOR I%=1 TO pgm%
1020 FOR J%=1 TO na%
1030 K%=FNloc(J%)
1040 I%?(&6F+K%)=I%
1050 I%?(&7F+K%)=0
1060 NEXT
1070 NEXT
1080 ENDPROC
1090 :
1100 DEF PROCout(i%,j%,b,a%)
1110 PROClimit(i%)
1120 IF w%=1 PROCpagin(a%)
1130 IF a%=1 X(b%,pos%)=b
1140 IF a%=2 Y(b%,pos%,j%)=b
1150 b%?(&7F+FNloc(a%))=1
1160 acc%(b%,a%)=acc%(b%,a%)+1
1170 ENDPROC
1180 :
1190 DEF FNin(i%,j%,a%)
1200 PROClimit(i%)
1210 IF w%=1 PROCpagin(a%)
1220 acc%(b%,a%)=acc%(b%,a%)+1
1230 IF a%=1 =X(b%,pos%)
1240 IF a%=2 =Y(b%,pos%,j%)
1250 :
1260 DEF PROClimit(n%)
1270 LOCAL i%,k%:i%=0:b%=0
1280 REPEAT:i%=i%+1
1290 k%=i%?(&6F+FNloc(a%))
1300 lr%=pgl%*(k%-1)+1:hr%=lr%+pgl%-1
1310 IF n%<lr% OR n%>hr% GOTO1330
1320 b%=i%:i%=pgm%
1330 UNTIL i%=pgm%
1340 IF b%=0 GOTO1360
1350 w%=0:pos%=n%-lr%+1:ENDPROC
1360 w%=1:pgm%=(n% DIV pgl%)+1
1370 IF (n% MOD pgl%)=0 pgm%=pgm%-1
1380 pos%=n%-pgl%*(pgm%-1)

```

```

1390 ENDPROC
1400 :
1410 DEF FNloc(t%)
1420 =(t%-1)*pgm%
1430 :
1440 DEF PROCpagin(r%)
1450 LOCAL i%,j%:b%=FNmin
1460 PROCpage
1470 IF b%?(&7F+FNloc(r%))=1 PROCpagout
(r%)
1480 pg$=STR$(pgr%)+STR$(r%)
1490 p=OPENUP(pg$)
1500 FOR i%=1 TO pgl%
1510 IF r%=1 INPUT#p,X(b%,i%):GOTO1550
1520 FOR j%=1 TO 3
1530 INPUT#p,Y(b%,i%,j%)
1540 NEXT
1550 NEXT
1560 CLOSE#p
1570 b%?(&6F+FNloc(r%))=pgr%
1580 ENDPROC
1590 :
1600 DEF PROCpagout(r%)
1610 LOCAL i%,j%
1620 pg$=STR$(b%?(&6F+FNloc(r%)))+STR$(
r%)
1630 q=OPENUP(pg$)
1640 FOR i%=1 TO pgl%
1650 IF r%=1 PRINT#q,X(b%,i%):GOTO1690
1660 FOR j%=1 TO 3
1670 PRINT#q,Y(b%,i%,j%)
1680 NEXT
1690 NEXT
1700 CLOSE#q
1710 b%?(&7F+FNloc(r%))=0
1720 acc%(b%,r%)=FNmax
1730 ENDPROC
1740 :
1750 DEF FNmin
1760 LOCAL k%:mn%=9000
1770 FOR k%=1 TO pgm%
1780 L%=acc%(k%,r%)
1790 IF L%<mn% mn%=L%:b%=k%
1800 NEXT
1810 =b%
1820 :
1830 DEF FNmax
1840 LOCAL k%:mx%=-9000
1850 FOR k%=1 TO pgm%
1860 L%=acc%(k%,r%)
1870 IF L%>mx% mx%=L%
1880 NEXT
1890 =mx%
1900 :
1910 DEF PROCpage
1920 FOR u%=&70 TO &70+2*pgm%-1
1930 PRINT ?u%;
1940 NEXT:PRINT
1950 ENDPROC

```

Coming soon...

the 15th show

where you'll find all the latest home,
education and business software

Plus...

all the new hardware add-ons that will expand
the power and versatility of the Electron,
BBC Micro, Master and Compact

Renold Building
UMIST
Sackville St
Manchester



Friday to Sunday
March 20 to 22

Doors open 10am
Close 6pm Friday
and Saturday,
4pm Sunday

SAVE
50p

Bring this coupon to the Show
to get 50p per person off the
normal admission price of £3
(adults), £2 (children).
Valid for up to four people.



10am-6pm, Friday, 20 March
10am-6pm, Saturday, 21 March
10am-4pm, Sunday, 22 March

Renold Building
UMIST, Manchester

50p
OFF

COMMAND — the ultimate communications ROM — for the BBC Micro & Master

Command is the ideal companion for the Beebug Magic Modem, and other BBC compatible modems.

Command is a very special command driven communications ROM, with a powerful extended instruction set. Because it may be command driven, it is exceptionally easy to link instructions together in Basic to build your own customised communications software.

● Viewdata Terminal

A full feature Prestel Terminal with pull-down help screen, real time clock and Epson compatible mode 7 screen dump.

● Text Terminal

Send files to a friend for the cost of a phone call, and access hundreds of Bulletin Boards throughout the country. Features include: 40/80 column operation, Xon/Xoff protocol, and pull-down status panel showing how Command is configured.

● Telephone Directory

This excellent facility allows you to save a name, number and modem configuration onto disc, for easy recall at a later date. No need to remember telephone numbers anymore.

● Viewdata Editor

A full feature text editor, with a full range of editing commands, pull-down help window, and a pixel editor.



Over 50 powerful commands, including

- | | |
|------------|--------------|
| ★ ANSWER | ★ BAND |
| ★ CALL | ★ CONNECT |
| ★ DIRECT | ★ DISCONNECT |
| ★ DISPLAY | ★ DOWNLOAD |
| ★ ECHON | ★ GRAB |
| ★ LISTEN | ★ PAUSE |
| ★ PRON | ★ PSCREEN |
| ★ RETRY | ★ RINGS |
| ★ SAY | ★ SDUMP |
| ★ SEND | ★ STANDARD |
| ★ STAT | ★ TEXT |
| ★ UPLOAD | ★ VEDIT |
| ★ VIEWDATA | ★ XON |

PRICE

£39.00

MEMBERS
PRICE

£29.25

Supplied on ROM with 76
page manual, and handy
function key strip.

NEW

PRINTWISE — professional looking documents with the minimum of fuss

The ideal companion for Wordwise, View and Inter-word

Printwise is a low cost publishing aid, and allows you to create professional looking magazines, leaflets, posters — the possibilities are endless. Also use it for 'near-letter quality' printout for correspondence. Simply take your text file, specify the font styles you require, and Printwise will do the rest. No programming skills are necessary.

PRICE	MEMBERS PRICE
£30.00	£22.50

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

A.STYLISH

! ' # \$ % & ' () * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z

B.BOOKTEX

! ' # \$ % & ' () * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z

C.ULTRA

! ' # \$ % & ' () * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z

SUPPLIED ON DISC
WITH 60 PAGE
MANUAL

Personal Ads

Replay system for Acorn 8271 DFS £20, Solidisk 32K sideways RAM £15, Acorn speech system (chips only) £5. All with relevant manuals. Phone (0773) 764380 after 6pm please.

Master compatible Superior Software discs: Repton 2, Repton 3, Citadel £8 each. Elite £10. BBC B Scrabble tape £5. Two Peartree Master ROM cartridges £5 each. All software in mint condition in original packing. Tel: (026378) 488.

BBC B with Watford 32K RAM and ATPL sideways ROM, UDM single/double density DFS complete with Toolkit and Disc Doctor ROMs - £300 ono. Tel: Wigan (0942) 715753.

Cumana CS400 switchable 40/80 track Disc Drive with incorporated power unit, hardly used £150 ono. Tel: 01-952 7244.

6502 second processor (requires DNFS) with HiBasic, HiView, 6502 Development Pack £125. 40 track 100K disc drive £60. Tel: (0256) 464889 evenings.

Owner upgrading. Software and games comprising Beebugsoft Toolkit Plus, Quickcalc, Frak, Scrabble, Ski Slalom, Ghouls. All games and Quickcalc cassette versions - £35. Tel: (0243) 820640.

Wanted: Microwriter MKIV or Quinkey, reasonably priced. Replies to C.P.Jamieson, 50 Hong Kong Workshop REME, Malaya Lines, Sek Kong, BFPO 1.

ATPL 'Sidewise' ROM/RAM board £25, Watford 32K shadow RAM board £50, Beebugsoft Toolkit £18, Chalice Master ROM £10, Elite (disc) £10, Aviator (disc) £10 and BCalc (disc) £5. All boxed complete with manuals. Reasonable offers accepted. Tel: (0753) 853193.

For Sale: ROMs £15 each: Spellcheck III, Sleuth, Exmon II, Toolkit Plus, View 2.1, Watford DFS, Gemini Datagem. Discs £10 each: The Hobbit, Gemini Beebcalc, Gemini Money Management, Kansas Personal Finance. Tapes £5 each: Monopoly, White Knight Mk 12. ATPL sideways ROM board plus 16K RAM £20. All with documentation. Tel: (0923) 21425.

Toolkit Plus, Help II, GXR, Scribecheck ROMs unregistered £10. Program Builder, Elite £5 plus postage. Tel: (0442) 67952 (Herts).

Tandy 4 colour printer/plotter (Oric printer), cable to connect to BBC, spare pens and paper. £50 ono. Tel: (04023) 77138 (Romford).

BBC B Software: Wordease, Quickcalc, Beta-Base, Dialog Home Accounts Manager, Fileman, Elite, Wordwise Plus Handbooks Beverley and Smith both with demo discs. All on 40 track. £45 the lot. Tel: 061-928 3571.

Single disc drive, double sided 40 track with own PSU. £75 plus carriage. Tel: (0246) 418215 evenings.

Morley Teletext Adaptor and V.2a ROM for sale, only £50 (includes p&p), complete with manual in original packaging. ROM clashes and loss of interest in Teletext reasons for sale. David Shepherdson, 3 Tarn Villas, Cowpasture Road, Ilkley, West Yorkshire LS29 8RH. Tel: (0943) 609866.

Beebugsoft Sleuth ROM £20. P.Jacobs, 15 Cheriton Road, Winchester, Hants SO22 5EQ.

Wordwise Plus £30, Spellcheck II (80 track dictionary disc) £15, Wordease (disc version) £10, or all 3 for £50. BEEBUG Toolkit ROM £10. ATPL ROM board and sideways RAM plus SMP Computing Sideways RAM utilities (e.g. print buffer), all for £25. Elite (disc version) £7.50. All originals and include manuals, p&p etc. Tel: S.Gaynor (0462) 682961.

40 track disc drive, mains powered, complete with cable £50; Wordwise Plus ROM and manuals etc £26. Vu-Type tape and manual (typing tutor) £6. Forth tape and book £9. Discmaster disc and manual £10. Book '30 Hour Basic' £5. Book 'Advanced User Guide' £6. Approx. 55 floppy discs £50. BEEBUG magazine April 1984 to present £10. Contact Mike on Bedford (0234) 218918.

6502 second processor £100, Aries B32 RAM board £60, ATPL Sidewise ROM board £20. Tel: (0782) 392713 evenings only.

Discs. Pack of 30 5.25" DS, brand new, in sleeves £20. T.Wiltshire, "Bramblings", Pelican Road, Pamber Heath, Basingstoke, Hants RG26 6EL. Tel: (0734) 701163.

Changing hobby. I have a BBC Model B (OS 1.2) with OPUS DFS and disc drive, shadow RAM, Wordwise Plus and 10 discs. All mint condition. Have you a Double Bass with bag and bow or £400? (preferably a Bass). Please phone (0629) 56771.

Acorn Z80 and 6502 second processors combined in one case with front switch and LEDs. Includes original software, manuals and other case. £250 ono. Acorn Prestel adaptor £75 ono. Wordstar professional for Acorn Z80 £225 ono. Acornsoft GXR ROM £18, Exmon II £15. Tel: (0533) 312661.

BBC bits. Nightingale modem, ATPL ROM board, two disc drives, MCP40 plotter, ROM and disc based software. Offers OR exchange for Opus 5802/Hard disc/Red boxes? Tel: Alan on 01-340 2646.

Watford 32K shadow RAM board, ROM and manual £40. Tel: (089582) 3613 evenings.

BBC B, Microvitec 1431 RGB monitor, 6502 second processor, DNFS kit, Acorn speech kit, cassette recorder, Acorn software and books - £550. Tel: Liverpool 051-264 8445.

Cumana CS100 drives (40 track SS with PSU). Pair on dual cable, plus manual, utilities disc, and spare cable. £140. Tel: North Weald (037 882) 2622 evenings.

BBC B, OS 1.2, Basic II with Watford DD D.F.S. £230. Opus Challenger with 256 solid disc RAM £130. ATPL ROM board, ROMs - Toolkit, Disc Doctor, Exmon II, Printmaster. All items complete with manuals. Tel: Twyford (0962) 713546.

Acorn 6502 second processor. As new with ROMs and manual - £99. D.Gibbons, Wantage (02357) 3253.

BBC B with Acorn 1.2 DFS, Double Sided 40/80 track 400K Viglen disc drive, ATPL Sidewise ROM board with sideways RAM, + £700 worth of software including Wordwise Plus, other ROMs and various games. Fitted with brand new keyboard: £650 ono. Tel: 01-444 7748.

Creative Sound (Acornsoft), Manual and two discs. Boxed £11. Murom (Beebugsoft), ROM and manual £15. R.J.Follett, 26 Arbor Lane, Winnersh, Berks RG11 5JD.

Diablo 1300 Heavy duty Daisywheel 40cps plus interface £180, ribbons extra. Tandon twin 40 track £110. Hitachi 9 inch monitor £25. Snap EV1 camera £60. ZX81 +16K offers. All vgc working. Tel: 01-851 7266.

Pace Nightingale Modem 300/300 and 1200/75 Baud with auto-dial emulation plus Commstar 1 ROM and 80 track disc software. All original manuals and user guide. Suitable for use with BBC micro and B.T. standard two wire telephone line. New in August 1986 cost £204, will accept £95 for quick sale. Telephone (0625) 27306. Macclesfield Cheshire area.

Toolkit Plus ROM £20, Sleuth 1.06 ROM £15. Both with manuals and reference cards. Also miscellaneous software. Tel: Doncaster (0302) 744005.

Star Gemini 10X printer (120 cps), manual and lead, little used £95. Voltmace 14/B1 adaptor box, keypad and joystick driver cassette, instructions £6. Tel: 01-943 6616 (work) or 01-948 3956 (home).

New Master Turbo, twin 40/80 DS drive, 3 reference manuals (Welcome + part 1 & 2 reference), 3 ROM cartridges, R/W ROM/RAM cartridge. Only £750 for over £900's worth plus lots of software. Tel: (0705) 256699 (Hampshire).

Acorn Z80 2nd processor, with all software and manuals, boxed as new £225 (will deliver). Tel: P.M.Brown (0955) 82623.

Wanted by Master owner - copy of the manual for Acornsoft's Graphics Extension ROM (to buy or borrow). Peter Jennings, 26 Rowan Close, St.Albans, Herts AL4 0ST. Tel: (0727) 61835.

UK Bulletin Boards

(Continued from Vol.5 No7.)

TBBS West Midlands 18.00-08.30	0384 635336 (London) 300/300	WBBS Wimbledon 19.00-02.00	01 542 3772 (London) 1200/75 & 300/300
Techno Fresh systems 24 Hours	0570 423082 (Dyfed) 300/300 1200/75	
Technoline 24 Hours	01 450 9764 (London) 1200/75 Viewdata	<u>New UK Bulletin Boards</u>	
Telemac 15, OBBS 24 Hours	0625 33703 300/300	HBBS No.1 24 Hours	0274 42957 (Bradford) 1200/75 300/300
Timestep (Suffolk)	0440 820002 1200/75	Infocom 24 Hours	021 476 9881 (Birmingham) 1200/75 300/300
TUG London 24 Hours	01 200 7577 (London) 300/300	Intaview 24 Hours	021 622 5010 (Birmingham) 1200/75
TUG Board II 20.00-08.00	021 444 1484 (Birmingham) 300/300	Madhouse 24 Hours	061 477 8405 (Manchester) 1200/75 300/300
WBBS Lowestoft 18.00-08.00	0502 515935 (Suffolk) 300/300	SBBS Stevenage 24 Hours (0800- 2300 ringback	0438 722128 (Stevenage) 300/300

Members Corner

Mr Luckhurst writes from Kingsdale School, a comprehensive in Dulwich, south London. The school has a blind student who started on GCSE exam courses September last, and although the school is comparatively well equipped with computer hardware and software, requires help from anyone willing to help in the conversion and writing of programs to help this student. If anyone can help, then please contact Mr Luckhurst at the school. The address is Kingsdale School, Alleyn Park, Dulwich, London SE21 8SQ, or telephone 01-670 7575.

David Else is interested in receiving weather facsimile transmissions from the Meteorological Office in Bracknell, using a Sony ICF 2001 receiver. If anyone can help David regarding interfacing and programming please write to him at 5 Copthorn Road, Upper Colwyn Bay, Clwyd LL28 5YP or phone 0492-531584.

Do you use Japanese Kanji on a BBC micro or Torch system? If so, Mr H. Williams of 15 High View Road, Onslow Village, Guildford GU 5RS would be interested to hear from you.

BEEBUG MAGAZINE OFFERS

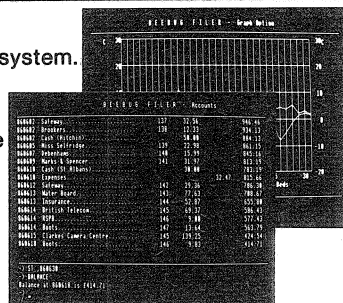
BEEBUG FILER — see this issue
BEEBUG magazine's own database management system.

Suite of five programs including:

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

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

Price £5.50 plus £1.00 post and packing.



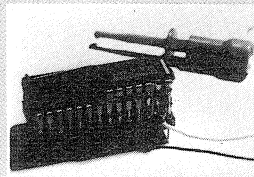
BEEBUG Sideways RAM Module — see Vol.5 No.7

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

- ★ Plugs directly into any BBC ROM socket
 - ★ Construction and software described in magazine
- Available as a kit of parts for you to make up yourself, or ready made

Price £8.95 (kit) £12.95 (ready made)

Post & packing £1.00 in each case

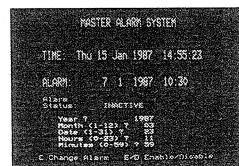


BEEBUG Master Alarm ROM — see this issue

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

Supplied on EPROM

Price £6.45 plus £1.00 post & packing



ORDER FORM

Name

Address

Membership Number

Signature

I enclose cheque for £

Please debit my Access/Visa No.

Please send:

Filer Disc £

BEEBUG Sideways RAM Kit £

BEEBUG Sideways RAM Module £

BEEBUG Master Alarm ROM £

Post & Packing £

Total £

--	--	--	--	--	--	--	--	--	--

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

Business Ads

MARKS AND STATISTICS For teachers and lecturers. Simple to use spreadsheet format. 330 students per file. Up to 300 subjects. Sorts, analyses, normalises, calculates means, applies weighting factors. Prints histograms and lists. Full documentation supplied. Disc (40 or 80 track) £17. In-Form, 73 Woodfield Park, Colinton, Edinburgh EH13 0RA.

Oscilloscope Teaching Simulator program, perfect for showing CRO operation. Demonstration and real-time modes, extremely versatile. Only £7. For details send large SAE to: Osc-Demo,

63 Stanbrook Road, Shirley, W.Midlands B90 4US.

Chess Expert System. Send for details and prices of this data capture and editing system with the gigantic database. Still available: Opening Knight and Knight Writer for White Knight Mk 12 and now Colossus 4. Large opening repertoire and game printing programs, many delighted users. Send for free information pack, or cheque for £7 to Bernard Hill, Hawthorn Bank, Scott's Place, Selkirk TD7 4DP.

Events

Electron and BBC UMIST, Manchester 20-22 Mar
Micro User Show (change of venue)

Electron and BBC Old Horticultural 8-10 May
Micro User Show Halls, Westminster

Acorn User Show Barbican Centre 23-26 Jul

PCW Show Olympia, London 3-8 Sep

Electron and BBC Old Horticultural 13-15 Nov
Micro User Show Halls, Westminster

New High Scores

Supplier	Game	Score	Player
ACORNSOFT	Carousel	98020	S.Clark
ALLIGATA	Blogger	23280	S.Clark
MELBOURNE	Exploding Fist	7th Dan	K.Brown
		41700	K.Brown
SUPERIOR SOFT	Repton 2	20468	I.Simmons
SUPERIOR SOFT	Space Pilot	112300	J.Benger
SUPERIOR SOFT	Stryker's Run	19200	M.Williams
US GOLD	Commando	436000	K.Brown



MASTER SERIES

Master add-ons review

The latest add-ons from Vine Micros and Pear- tree Computers for the Master 128 reviewed by Peter Rochford.

Not all of the software in the Master's 128K ROM appeals to all users. Vine Micros, realising this, have released their Master ROM Overlay Board at a cost of £19.95. This allows paging out of any of the software contained in the

substitution of your own ROMs. The board comes with good documentation and takes just minutes to fit. After removal of the 128K ROM from its socket in the Master (Beware! Some Masters have their 128K ROM soldered in), the board plugs into the vacated socket, and the 128K ROM is inserted into a socket on the Vine board.

Product : ROM Overlay Board

Supplier: Vine Micros
Marshborough,
Sandwich,
Kent CT13 0PG.
Tel. (0304) 812276

Price : £19.95 inc VAT & p&p

Three further sockets on the board are used for your ROMs, and by means of links, you can page out up to three of the eight ROMs in the Master's 128K ROM.

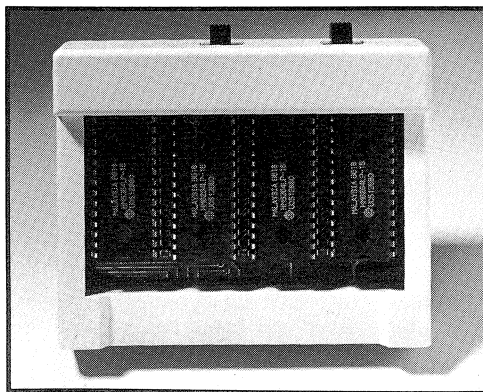
There are slight problems with this though, as View in the 128K ROM contains some OS code as does Terminal, whilst the DFS contains the SWR routines. This leaves the ADFS, Edit, Basic and ViewSheet as possible candidates for paging out without trouble. I find the board a useful addition to my Master even so.

Product : ROM Cartridges

Supplier: Peartree Computers Ltd
St. George's House,
14 George St,
Huntingdon,
Cambs PE18 6BD
Tel. (0480) 50595

Price : £8.35, £10.65, £12.95.
With RAM £33.50.
All inc VAT & p&p.

One way around the lack of ROM sockets on the Master is by the use of cartridges. The Acorn ones are hideously expensive at £14.95, and easily broken when removing ROMs. Peartree Computers have now produced several cartridges of their own for the Master. The cheapest at £5.95 consists of a plug-in PCB with a handle for removal, but no case. This is fine, but does wobble about when in the machine, though this caused no problems in use. Their next offering is a cased version called Pear 1 at £7.95, again allowing the use of two ROMs and worth the extra I feel, for the better fit and protection.



Peartree's MR6000 cartridge costs £9.95, and will accommodate up to four ROMs. Any two ROMs can be active at one time, and are selected by means of a switch on top of the cartridge. A further feature of the cartridge is that it may be used as two 16K banks of SWR by inserting four 8K CMOS RAM chips. The RAM may be loaded like any of the Master's own SWR banks with ROM images, and a switch on the cartridge allows write-protect. At the price, this is a real gem, though one of the units we tested did need a file applying to it before we could plug it in. Peartree also supply the cartridge with RAM as the MR7200 for £26.95 along with a useful utility disc.

CONCLUSION

All these products can be recommended without hesitation to those who have a need for the facilities they provide. Construction and design in all cases is good, and the Peartree cartridges in particular, represent excellent value for money.



MASTER SERIES

A versatile auto-alarm system



David Graham develops an auto-alarm system for the Master 128 which will remind you of important dates and times, even if your machine is switched off for most of the intervening period.

The combination on the Master of continuously running clock calendar and battery-backed CMOS RAM opens up the possibility of producing an alarm system to remind you of forthcoming events, which is not upset by switching off the computer. In this article, we will develop software to implement this idea. This deals with a central machine code program which is capable of continuously monitoring the time and date supplied by the Master's calendar, and comparing it with an alarm time (in years, months, days, hours and minutes) stored in compressed form in four bytes of CMOS RAM. When the dates and times match, an alarm is sounded every ten seconds.

Accompanying this assembler program is a simple alarm management program written in Basic, which allows easy control of the alarm system. This could be extended to permit a whole calendar of events to be catered for, so as to prompt you of each important event minute-by-minute until the end of the century. But we begin by taking a closer look at the alarm system itself.

THE ALARM SYSTEM

This has been designed to reside in sideways RAM so that it can operate invisibly along-side other programs; and the code provided may also be blown directly onto EPROM to allow the system to work completely independently of the user, even if the computer is switched off for long intervening periods. If you do not have a blower, a ready-blown ROM may be obtained from us - see details at the end of this article.

The software makes use of just two star commands:

- *A1 Activate System
- *A0 Disable System

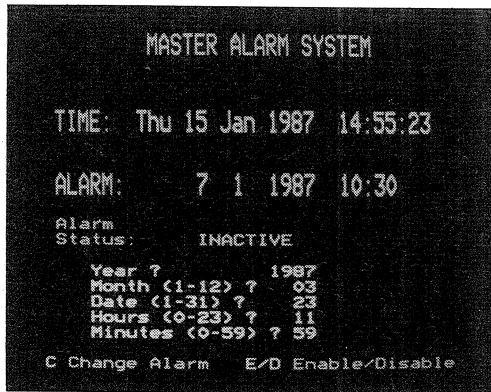
Once the alarm is active, a comparison is made once every second between the compressed alarm data held in CMOS RAM, and the current Master clock setting. If the alarm time has been reached, the alarm sounds every ten seconds until either a new alarm time is entered, or until the command *A0 is issued.

To test out the system, type in the assembler listing, and save it away before running it. When it is run, you will be prompted to press f8 to save the assembled code, and then f9 to load it into sideways RAM. Once you have done this, press Ctrl-Break to initialise the ROM image, and then type *A1. The command should be accepted, though without visible response. Now if you type:

*FX162,43,255

you should hear the alarm ring. Also, if Break is pressed at any time when the system is active, the message:

Alarm System Active
will appear.



ALARM MANAGEMENT SYSTEM

Once the assembler routine appears to be working, type in the Basic listing, and run it. You should see a continuously updated display of the clock, together with the current alarm setting, which may initially look rather odd. Press "C" to enter a new alarm setting. If any field is to remain unchanged, simply press Return at the appropriate point. After you have done this, you should see your new alarm time displayed. This screen also displays the status of the alarm system, and after the time change option, you will see that it reads "Active". Note that "E" and "D" can be used to manually enable or disable the system.

The alarm will cope with any time setting from 1985 to 1999, and you should find that the software will operate invisibly under all circumstances. It is unaffected by pressing Break or Ctrl-Break; and if present on ROM, it is even unaffected by turning off the computer. As soon as you switch on again, it will continue to operate as before. If it is in sideways RAM, you will need to reload it when the machine is powered up. To do this, type:

```
*SRLOAD romcode 8000 n
```

where n is the RAM bank number (4 - 7). Then press Ctrl-Break to initialise the ROM image.

You now have a most useful working alarm system for your Master, that fully exploits some of its special features. The system described could in turn be readily extended to cover a whole calendar of future events.

MASTER ALARM ROM

An enhanced version of this software is available in ROM which contains a third star command (*AD) to download and run the Alarm Management program, making full alarm control instantaneously available. The Alarm ROM costs £6.45 + £1.00 pp (inc. VAT), and is available from our St Albans mail order dept. Call in for a demonstration.

TECHNICAL DETAILS

CMOS USAGE

CMOS RAM No	Function	Range
40 high	Year	0 - 14 (1985-1999)
40 low	Month	1 - 12
41	Date	1 - 31
42	Hours	0 - 23
43 bits 0-5	Mins	0 - 59
43 bit7	Alarm active flag	
43 bit6	Alarm ringing flag	

ZERO PAGE

Locations &70 - &7D are used as temporary storage, but are all preserved.

SOUND CHANNEL

Channel 2 is used for the alarm (since 1 is commonly used, and 3 is used by VDU7). To alter the volume of the alarm, change the value &F9 set in line 1110. &F1 gives max volume, &FF minimum.

```
10 REM Program Master Alarm System
20 REM Version B 0.42
30 REM Author David Graham
40 REM Beebug March 1987
50 REM Program subject to copyright
60 :
70 OSASCII=&FFE3:OSNEWL=&FFE7
80 OSBYTE=&FFF4:OSWORD=&FFF1
90 OSC=&FFF7:caddr=&F2
100 version=4:version$=" 0.42"
110 chr=&70:clock=chr+6:T=clock
120 DIM R% 3000
130 FOR pass=0 TO 1
140 O%=R%:P%=&8000
150 [:OPT pass*3+4
160 \***** ROM Header
170 .start:BRK:BRK:BRK
180 .service:JMP servicentry
190 EQUB &82
200 EQUB copyright-start
210 EQUB version
220 .title
230 EQUB "Master Clock System"
240 BRK:EQUB version$
250 .copyright:BRK
260 EQUB "(C) David Graham 1987":BRK
270 .servicentry:PHP:CMP #&27
280 BNE se0:JMP breakcode
290 .se0:CMP #4:BNE sel:JMP command
300 .sel:PLP:RTS
310 .end:PLY:PLX:PLA:PLP:RTS
320 .endA:PLY:PLX:PLA:LDA #0:PLP:RTS
330 \***** Respond to Break
340 .breakcode:PHA:PHX:PHY
350 JSR testbit:BPL break1:JSR pushzp
360 JSR brkness:JSR setvector
370 LDA #0:JSR settimer:LDA #14
380 LDX #5:JSR OSBYTE:JSR pullzp
390 .break1:JMP end
400 .setvector
410 LDA #&30:STA &220
420 LDA #&FF:STA &221
430 LDA #handler MOD 256
440 STA &D9F+48
450 LDA #handler DIV 256
460 STA &D9F+49:LDA &F4
470 STA &D9F+50:RTS
480 .testbit:LDX #43:LDA #161
490 JSR OSBYTE:TYA:RTS
500 .brkness:LDX #0
510 .bme1 LDA bme2,X
520 JSR OSASCII:INX:CMP #13
530 BNE bme1:JSR OSNEWL:RTS
540 .bme2:EQUB "Alarm Active":EQUB 13
550 \***** Interrupt Handler
560 .handler:PHP:PHA:PHX:PHY
570 CLD:JSR pushzp
580 LDX #43:LDA #161:JSR OSBYTE
590 TYA:ASL A:BCC out1:ASL A
600 BCC nonring:JSR beeper:LDA #&80
```



```
Acorn MOS
Alarm Active
Acorn 1770 DFS
Basic
>_
```

```
610 BRA out2
620 .nonring JSR readclock
630 JSR convert
640 JSR readmos
650 JSR compare
660 BCC out1
670 LDA #&40:STA chr:JSR flagchange
680 .out1 LDA #0
690 .out2 JSR settimer:JSR pullzp
700 PLY:PLX:PLA:PLP:RTS
710 \***** Clock Read
720 .readclock:LDA #1
730 STA clock:LDA #14
740 LDX #clock MOD &100
750 LDY #clock DIV &100
760 JSR OSWORD:RTS
770 \***** BCD to Hex
780 .bdec:TAY
790 AND #&F:PHA:TYA:AND #&F0:STA chr
800 CLC:LSR chr:PLA:ADC chr:LSR chr
810 LSR chr:ADC chr:STA chr:RTS
820 \***** Clock Converter
830 .convert:LDA T:JSR bdec:SEC
840 SBC #85:ASL A:ASL A
850 ASL A:CLC:STA T:LDA T+1:JSR bdec
860 ADC T:STA T
870 LDA T+2:JSR bdec:STA T+1
880 LDA T+4:JSR bdec:STA T+2
890 LDA T+5:JSR bdec:STA T+3:RTS
900 \***** Read CMOS RAM
910 .readmos:LDX #43
920 .rc1 LDA #161:STX chr:JSR OSBYTE
930 DEC chr:LDX chr:CPX #42:BNE rc2
940 TYA:AND #&3F:TAY
950 .rc2 STY chr -38,X:CPX #39
960 BNE rc1:RTS
970 \***** Compare Clock and Alarm
980 .compare:LDX #0
990 .cm1 LDA chr+1,X \ Alarm settings
1000 CMP clock,X \ Clock Settings
1010 BEQ cmcont:BCC cmring
1020 CLC:BRA cmend
1030 .cmcont INX
1040 CPX #4:BNE cm1
1050 .cmring SEC
1060 .cmend RTS
1070 \***** 2 Tone Beeper
1080 .beeper
1090 LDA #21:LDX #6:JSR OSBYTE
1100 LDA #2:STA T:LDA #0:STA T+1
1110 LDA #&F9:STA T+2:LDA #&FF
1120 STA T+3:LDA #100:STA T+4
```

```
1130 LDA #5:STA T+6
1140 LDA #5:STA T+5
1150 .sn1 LDX #clock MOD &100
1160 LDY #clock DIV &100
1170 LDA #7:JSR OSWORD:LDA T+4
1180 EOR #&70:STA T+4:DEC T+5
1190 BNE sn1:RTS
1200 \***** Set Timer
1210 .settimer:ASL A:BCC til:ASL A
1220 BCC ti2:LDA #&90:LDX #&E8
1230 BRA callset
1240 .ti2 LDA #&18:LDX #&FC:BRA callset
1250 .ti1 LDA #&9C:LDX #&FF
1260 .callset
1270 STA clock:STX clock+1:LDA #&FF
1280 STA clock+2:STA clock+3
1290 STA clock+4
1300 LDX #clock AND &FF
1310 LDY #clock DIV &100
1320 LDA #4:JSR OSWORD:RTS
1330 \***** Change Flags
1340 .flagchange:LDA #161
1350 LDX #43:JSR OSBYTE:TYA:LDX chr
1360 BNE flag1:AND #&3F:BRA flag2
1370 .flag1:ORA chr
1380 .flag2:TAY:LDX #43:LDA #162
1390 JSR OSBYTE:RTS
1400 \ ***** Save Zero Page
1410 .pushzp:LDX #&FF:pzp1 INX:LDA chr
,X:STA &100,X:CPX #&D:BNE pzp1:RTS
1420 .pullzp LDX #&FF:pzp2 INX:LDA &100,X:STA chr,X:CPX #&D:BNE pzp2:RTS
1430 \***** Command Interpreter
1440 .command:PHA:PHX:PHY
1450 LDA (caddr),Y:CMP #ASC("A")
1460 BNE comm1:INY:INY:LDA (caddr),Y
1470 CMP #&D:BNE comm1:DEY
1480 LDA (caddr),Y:CMP #ASC("I")
1490 BEQ install:CMP #ASC("0")
1500 BEQ remove:.comm1:JMP end
1510 \***** Install Alarm
1520 .install
1530 JSR pushzp:LDA #&80:STA chr
1540 JSR flagchange:JSR setvector
1550 LDA #14:LDX #5:JSR OSBYTE
1560 LDA #0:JSR settimer
1570 JSR pullzp:JMP endA
1580 \***** Remove Alarm
1590 .remove
1600 JSR pushzp:STZ chr:JSR flagchange
1610 LDA #13:LDX #5:JSR OSBYTE
1620 JSR pullzp:JMP endA
1630 ]:NEXT
1640 PRINT"f8 to save"
1650 PRINT"f9 to reload"
1660 *K.8 OS. ("S.romcode "+STR$~R%+" "+
STR$~O%+" 0 FFFF8000")|M
1670 *K.9 *SRLOAD romcode 8000 4|M
*****
```



```

10 REM Program Alarm Control System
20 REM Version B 0.4L
30 REM Author David Graham
40 REM Beebug March 1987
50 REM Program subject to copyright
60 :
100 DIM alarm(5):ans=0:@%=2
110 REPEAT
120 ON ERROR GOTO 1580
130 MODE7:VDU23,1|
140 PROCinit
150 REPEAT
160 PROCclock
170 PROCread
180 key=INKEY(0)
190 UNTIL key>66 AND key<70
200 *FX15,1
210 ON ERROR IF ERR=17 THEN GOTO 110 E
LSE GOTO 1580
220 SOUND2,-15,10,2
230 IF key=67 PROCinput:PROCreset
240 IF key=68 OSCLI("A0")
250 IF key=69 OSCLI("A1")
260 UNTIL FALSE
270 :
1000 DEFPROCclock
1010 pad2$=MID$(TIME$,5,1):IF pad2$="0"
pad2$=" "
1020 clockdisp$=LEFT$(TIME$,3)+" "+pad2
$+MID$(TIME$,6,10)+" "+RIGHT$(TIME$,8)
1030 FOR A=1 TO 2
1040 PRINTTAB(9,A+6)clockdisp$
1050 NEXT:ENDPROC
1060 :
1070 DEFPROCread:PROCreadout
1080 IF alarm(5)<10 pad$="0" ELSE pad$=
""
1090 FOR A=1 TO 2
1100 PRINTTAB(13,A+10),alarm(3)" "alarm
(2)
1110 PRINTTAB(20,A+10)"19";alarm(1);" "
,alarm(4);":":pad$;alarm(5)
1120 NEXT
1130 PRINTTAB(14,15);:IF bit1 PRINT"ACT
IVE";SPC2 ELSE PRINT"INACTIVE"
1140 PRINTTAB(25,15);:IF bit1+bit2=192
PRINTCHR$136;"RINGING" ELSE PRINTSPC8
1150 ENDPROC
1160 :
1170 DEFPROCreadout
1180 FORA=1 TO 4
1190 Z=FNreadc(A+39)
1200 IF A=1 alarm(1)=(Z DIV 16)+85:Z=Z
MOD 16
1210 IF A=4 bit1=Z AND 128:bit2=Z AND 6
4:Z=Z AND &3F
1220 alarm(A+1)=Z
1230 NEXT:ENDPROC
1240 :

```

```

1250 DEFFNreadc(X%):A%=161:=(USR(&FFF4)
AND&FF0000)/DIV&10000
1260 :
1270 DEFPROCinput
1280 VDU23,1,1|:RESTORE
1290 FOR A=1 TO 5
1300 READ A$,L1,L2
1310 REPEAT
1320 B$="":IF alarm(A)<10 B$="0"
1330 PRINTTAB(5,16+A)A$;B$;alarm(A);
1340 VDU8,8
1350 INPUT""ans$:ans=VAL(ans$)
1360 UNTIL ans$="" OR (ans>L1 AND ans<L
2)
1370 IF ans$<>"" THEN alarm(A)=ans
1380 NEXT:ENDPROC
1390 :
1400 DATA"Year ? 19",84,100,"Mo
nth (1-12) ? ",0,13,"Date (1-31) ?
",0,32
1410 DATA"Hours (0-23) ? ",-1,24,"Min
utes (0-59) ? ",-1,60
1420 :
1430 DEFPROCinit
1440 FOR A=1 TO 2
1450 PRINTTAB(8,1+A)CHR$134CHR$141"MAST
ER ALARM SYSTEM"
1460 NEXT
1470 FOR A=1 TO 2
1480 PRINTTAB(0,6+A)CHR$134CHR$141"TIME
":CHR$131
1490 NEXT
1500 FOR A=1 TO 2
1510 PRINTTAB(0,10+A)CHR$134CHR$141"ALA
RM":CHR$131
1520 NEXT
1530 PRINTTAB(1,14)CHR$134"Alarm"
1540 PRINTTAB(1,15)CHR$134"Status:":CHR
$131
1550 PRINTTAB(0,23)CHR$131"C"CHR$134"Ch
ange Alarm"SPC2CHR$131"E/D"CHR$134"Enabl
e/Disable"
1560 ENDPROC
1570 :
1580 @%=10:MODE7:REPORT
1590 PRINT" at line ";ERL:END
1600 :
1610 DEFPROCreset
1620 PROCwrite((alarm(1)-85)*16+alarm(2
),40)
1630 PROCwrite(alarm(3),41)
1640 PROCwrite(alarm(4),42)
1650 PROCwrite(alarm(5),43):OSCLI("A1")
1660 ENDPROC
1670 :
1680 DEFPROCwrite(n,X)
1690 OSCLI("FX162,"+STR$(X)+",""+STR$(n)
)
1700 ENDPROC

```



MASTER SERIES

Master Hints

Moving around within the ADFS directory structure can be a bit tedious at times. Here are some hints to speed things up a little.

Don't Nest Too Deep

This one is pretty obvious. The deeper you nest, the longer it will take you to specify a route from any directory to any other. For many purposes, you will find that one level of nesting is enough. This gives you up to 47 possible directories as immediate sub-directories of the root directory.

Use Wild Cards

When specifying a route to a given file, or when specifying a pathname for *DIR or *CDIR, enormous savings can be made by the use of wildcards. For example:

```
*DIR $.JANWORK.SPREADSHEET.MAINFILES
```

could be reduced to:

```
*DIR $.J*.S*.M*
```

But beware. If the directory name specification is not unique, the ADFS will take the very first match in ASCII order. So if your disc also contained the directory \$.J1, the ADFS would select the wrong route. To avoid this confusion, you could use:

```
*DIR $.JA*.S*.M*
```

To increase the effectiveness of wildcards on the ADFS it is worth making all directory names differ from each other as early as possible in the word.

Using Function Keys

One of the fastest ways to move around an ADFS disc is by using the function keys. The key definitions can be stored in an EXEC file, and with a little care, can be made to operate just as well from within Wordwise, View or Viewsheet as from immediate mode Basic. For example:

```
*KEY0 *CAT|M
*KEY1 *DIR :0.$.FIRST|M
*KEY2 *DIR :0.$.SECOND|M
```

and so on. Here, key f0 is used to catalogue the currently selected directory, and the remainder of the keys are used to move to various specified directories. You may also find it useful to have each key actually catalogue the directory which it selects, as follows:

```
*KEY1 *DIR :0.$.FIRST|M*.*M
```

This method ensures that the key may still be used from within Wordwise and the View family, and even the Editor, provided that you are in Command Line mode and have enabled the function keys with *FX228,1.

Function Key Doubling

If you wish to access many different directory names from the function keys, you will soon run out of room at around 10. But by a bit of subtlety you can double up on their effect. The following definition illustrates the principle:

```
*KEY0 IF INKEY(-97) OS.("DIR $.FIRST")
ELSE OS.("DIR $.SECOND")|M
```

The effect of this is that if you press f0 on its own you will be taken to \$.FIRST, but if you press Tab-f0 (in the same way that you might press Shift-f0) you will be taken to \$.SECOND. This immediately doubles the capacity of the function keys, and helps capitalise on the increased function key buffer allocation on the Master and Compact. Next month we will take this idea one stage further.

Extra Key Strips

One of the snags of using the function keys is that you need to keep changing the keystrips. The layout of the Master allows a new type of keystrip to be used in addition to the normal variety. You will find that a second keystrip can be fitted between the top of the function keys and the base of the perspex key strip retainer. It must of course be narrower than the normal strip, but may nevertheless hold the legends for up to twenty keys. If you make the strip from thin card, and give it a flange which drops down to the left of f0 and to the right of f9, it will remain in place without the necessity of fixing.

Filer Database System

The Last Word

Mike Williams describes some further enhancements and improvements to our popular Filer series of programs, and it's all available on one disc too!

This month, we bring to a close (for the time being at least) the various articles and programs forming the BEEBUG Filer Database Management System. This series, which commenced in November 1985 (BEEBUG Vol.4 No.6) has proved very popular, and many and varied are the applications to which readers have put the various programs. Now that we can afford to look back, there are a number of minor improvements that can be made, together with suggestions or other information to help you make the most of these programs.

It is impossible to include all the Filer programs on the monthly magazine cassette/disc, so we are providing a separate Filer disc (sorry - no cassette). This contains all the Filer programs, updated and revised as described here. The disc also contains a front-end menu to the Filer programs, and the individual programs are tailored to work with this.

The programs are supplied complete with a 'Help' system so that all commands can be rapidly described on-screen for quick reference. As space is at a premium in the programs, a number of revisions have been incorporated. The versions on the disc are therefore not identical to the original versions, but all line numbering, variable, function and procedure names have all been kept as close as possible to the originals, and all amendments and additions listed here can be made to your existing programs without further ado.

The Filer disc is accompanied by a revised and expanded version of the Filer Notes available previously. These now cover all the programs in the Filer Database Management system. The whole is

available now at £5.50 plus 50p p & p direct from BEEBUG (see supplement).

FILER - THE DATABASE PROGRAM

(Vol.4 No.6, 7, 8, updates Vol.5 No.1, 2)

Space is already at a premium with this program. You can make more space available by omitting the sort routine, now that a separate sort program is available, or any other option you do not need (mail-merge for example). If you do delete any command, make sure that the number of commands is changed (at line 1100), and that the test values from line 1630 for a continuous sequence.

Alternatively, compressing and compacting the program, maybe by using suitable commands in Toolkit or similar, will also help to make more space available. When adding any new feature, do check to make sure it is operating correctly before moving on to any further additions. In this way you should be able to locate quickly any typing errors you may make.

Some people have found it inconvenient that the ADD command allows only one record at a time to be added to a file. These two additional lines allow as many records as you like to be entered following one ADD command. Terminate record entry by pressing Escape.

```
8090 REPEAT
```

```
8230 UNTIL flag>0
```

Another inconvenience is that the DISPLAY and PRINT commands allow either one or all records to be specified. The following additions allow any continuous range of records to be selected with these commands in the form:

```
DISPLAY n1,n2
```

```
PRINT n1,n2
```

where n1 is the first record and n2 is the last record to be displayed or printed. You can still use the SELECT command to select records from within a range.

```
4820 LOCAL G,I,n1,n2,flag2
```

```
4870 n1=1:n2=rec-1:IF flag<2 flag2=FNlimits(p$) ELSE F1=OPENUP(p$):IF F1=0 PRINT "No such text":ENDPROC
```

```
4880 IF NOT flag2 PRINT "No such record (s)":ENDPROC
```

```
4940 FOR I=n1 TO n2
```

```
4980 IF I<n2 AND flag<>1 THEN G=GET
```

```
21800 DEF FNlimits(p$)
```

```
21840 IF p$="" =TRUE
```

FILER COMMANDS

GENERAL

COMMANDS	List valid commands
OPEN	Open data file for use
CLOSE	Close data file
END	End execution (return to menu)

FILER DATABASE

CREATE	Create data file or format file
ADD	Add new records to file
DELETE	Delete records from file
DISPLAY	Display records in file
EXTEND	Extend length of data file
FORMAT	Load print format
MARK	Specify marker for mail-merge
MERGE	Merge records with standard text
PRINT	Print formatted records
RENAME	Change a field name
SELECT	Select specified records
SHOW	Display details of file
SORT	Sort file
UPDATE	Update records in file

FILER GRAPHICS

GFORMAT	Load a graph format
GRAPH	Display a graph
PRINT	Print displayed graph
TITLE	Specify title for display

FILER ACCOUNTS

BALANCE	Give latest balance
CF	Carry forward to new file
DATE	Specify date for statement
STATEMENT	Prepare statement of account
TITLE	Specify title for statement

```

21860 I=INSTR(p$,","):IF I=0 n1=VAL(p$)
:n2=n1 ELSE n1=VAL(LEFT$(p$,I-1)):n2=VAL
(MID$(p$,I+1))
21880 IF n1>0 AND n2>0 AND n1<rec AND n
2<rec =TRUE ELSE =FALSE
21900 :
```

One feature of Filer that has caused many enquiries is the use of 'format' files. A format file is a special one-record file that describes how the main data records are to be printed. It is essential when using format files (and the same is true with graph formats) that the names of data fields are precisely the same as defined for the data file, and as used in the format description. The following additions provide a new command RENAME that allows the name of a data field to be changed. This provides a way of changing the name of a field without

having to delete the entire file and retype all the data.

Add RENAME at the end of the list of commands (line 1170) and increase the value at line 1100 by one. Then add the following lines (assuming this is the 17th command) and delete line 21640:

```

1790 IF C=17 PROCrename(pm$)

11000 DEF PROCrename(p$)
11020 LOCAL p,pl$
11040 IF NOT open% PRINT"No file open":E
NDPROC
11060 p$=p$+",":p=INSTR(p$,",")
11080 pl$=LEFT$(p$,p-1):p$=MID$(p$,p+1)
11100 p=INSTR(p$,","):pl=FNfield1(pl$)
11120 IF p*pl=0 PRINT"Invalid fields":EN
DPROC
11130 field$(pl)=LEFT$(p$,p-1)
11160 PTR#F=5+19*p1:PRINT#F,LEFT$(field$
(pl)+STRING$(12,"."),12)
11180 ENDPROC
11200 :
```

```

21600 DEF FNfield(p$,R)
21620 IF FNfield1(p$)>0 =record$(FNfield
1(p$),R) ELSE = " "
21660 :
21700 DEF FNfield1(p$):LOCAL I:I=0
21720 REPEAT:I=I+1:UNTIL p$=field$(I) OR
I=f
21740 IF I=f AND p$<>field$(I) =0 ELSE =
I
21760 :
```

The format of the new command is:

```
RENAME <field1>,<field2>
```

where field1 is the existing name, and field2 is the new name. It is not feasible, without considerable reorganisation, to allow either field widths or the number of fields to be changed after a file has been defined.

Another useful command that may be added is:

```
SHOW <filename>
```

This will display the current file description for reference. Add the new command to the end of the command list, and increase the number of commands by one, as before. Then add the following lines:

```

1800 IF C=18 PROCshow(pm$)

11400 DEF PROCshow(p$):LOCAL I
11460 IF open% IF F$<>p$ PROCopen(p$) EL
SE IF NOT open% PROCopen(p$)
```

```

11470 IF NOT open% ENDPROC
11475 PROCwindow1:VDU28,0,19,79,4,12,14
11500 PRINTTAB(5,2)"Length of file (records)";recn
11520 PRINTTAB(45,2)"Number of fields:";f
11530 PRINTTAB(5,4)"Field"TAB(11)"Fieldname"TAB(30)"Fieldwidth"
11540 FOR I=1 TO f
11560 PRINTTAB(5,5+I)"Field";STR$(I)":";
11580 PRINTTAB(16,5+I)field$(I)TAB(30,5+I)width$(I)
11600 NEXT I:VDU15
11630 PROCwindow2:PRINT"Press any key to continue":I=GET:PROChdr
11640 ENDPROC
11660 :

```

Thus SHOW will display the current field names (and field lengths) of a given file; RENAME can then be used to change one or more field names, and SHOW used to verify the change.

FILER - THE GRAPHICS PROGRAM (Vol.5 No.2)

This program which runs in mode 0 has no space left for any additions. However, the program is already quite complete. Three lines in the original listing (copied from Filer) should be corrected as shown so that the screen display is correct for mode 0:

```

1360 PRINTTAB(0,27)STRING$(80," ")
20420 VDU28,0,26,79,3:w=1:ENDPROC
20440 DEF PROCwindow2:VDU28,0,31,79,28

```

FILER - THE ACCOUNTS PROGRAM (Vol.5 Nos.4 & 5)

This addition to the FILER series has provoked considerable interest and several queries. One essential point is that all the entries in the main data file should be in date order. Use the SORT command in Filer, or the separate sort program, to re-order the file if necessary, otherwise unexpected results may occur.

An understanding of how the Accounts program handles the 'balance' will also help you to get more from the program. Whenever a statement (or balance) is called for, the program calculates the balance corresponding to each record up to the last within the period specified. The last balance calculated, the date and reference of the corresponding record are stored in the payments file. When new records are added and an updated balance

requested, calculation is only performed from the date of the last balance.

Should you have cause to edit or re-order any previous records, the program will be unaware of this, and a wrong balance will now be used for further calculations, giving incorrect results. The problem can be overcome by requesting a statement for a short period terminating earlier than any of the records that have been changed or re-ordered, and then requesting a new statement from that date. In this way the Accounts program can be forced to re-calculate balances as needed.

The program also contains a code number (123 at line 3480). Temporarily changing this to any other value will cause a re-initialisation of the payments file to take place. No information is lost, but ALL balances will subsequently be re-calculated.

The only real problem that has arisen with the Accounts program is with regard to accuracy; though this is obviously crucial. In the program all money sums are multiplied by 100 to avoid any errors that may arise through treating pence as a fraction of a pound. Although the arithmetic does work correctly, the subsequent string conversion to prepare the statement entries still occasionally falls foul of rounding errors. The following changes will cure this problem by a stricter use of integer variables and modified string conversion routine.

```

3520 PTR#F1=FDR-25:INPUT#F1,recb,p1$,p2
$:db=VAL(p1$):balance%=FNint(VAL(p2$))
4060 PRINT"Balance at ";STR$(db); " is £
";STR$(balance% DIV 100); ".";RIGHT$(STR$(
balance%),2)
4600 IF r>recb balance%=balance%-FNint(
FNval(amnt)):p$=STR$(balance% DIV 100)+"
."+RIGHT$(STR$(balance%),2):record$(bal,
0)=p$+STRING$(width$(bal)-LEN(p$),"."):P
ROCWrite(r,F,0) ELSE balance%=FNint(FNval
l(bal))
4760 recb=r-1:db=FNval(date):balance%=F
Nint(FNval(bal)):PROCwindow2:ENDPROC
7820 PROCread(pay(pay,1),F1,1):balance%
=balance%-FNint(VAL(record$(2,1)))
7840 IF p2$ PROCprintline(STR$(pay(pay,
0)),record$(1,1)," * ",FNstrip(record$(2,
1),"."),STR$(balance% DIV 100)+"."+RIGHT
$(STR$(balance%),2))
21900 DEF FNint(p)=INT(100*p+0.1)
21920 :

```

Recursion can result in some very concise programming when used properly. Surac shows how to make good use of this technique.

Recursion is a computing technique with something of a black magic image. However, there are a number of jobs that it's actually very useful for, as you'll see.

WHAT IS RECURSION?

Recursion is a programming technique in which a portion of code (e.g. a procedure or a function) performs a task by successively calling itself until some terminating condition is reached. Each time that the routine is entered, it solves part of the problem; it then checks to see whether it has finished. If it has more to do, it calls itself again; otherwise, it exits. When a call to a routine finally does exit, the whole nested chain unravels and the job is done.

Let's look at an example - see Procedure 1. This is a dummy program to handle a common job, but don't try running it! I hope that it's easy to see how it works; it makes you keep biting the wretched thing, each time with a new call, until it is finished, and then all the ENDPROCs are executed and the chain unravels.

That example also shows one of the reasons why recursion is not used a lot - there are very few cases where it is the only way of solving a problem. Almost always, the job can be done by a loop of some sort - see

Procedure 2. This is easier to understand than the recursive version, and would probably run faster into the bargain.

Procedure 1

```
1000 DEF PROC EatButty
1010 BITE Sandwich
1020 IF (Sandwich NOT Finished)
      THEN PROC EatButty
1030 ENDPROC
```

Procedure 2

```
2000 DEF PROC EatButty
2010 REPEAT
2020     BITE Sandwich
2030     UNTIL Sandwich Finished
2040 ENDPROC
```

Why bother at all, then? Programming languages should allow you to create programs which do the job YOU want to do in the way that YOU think of it but, as we'll see, some problems naturally lead to a recursive solution. Although they could be solved by loops, that is not the way we think of them.

WHAT DO YOU NEED FOR RECURSION?

Some computer languages (e.g. Logo and, to a lesser extent, Forth) almost insist that you use recursion. Others make it virtually impossible. BBC Basic makes it easy but optional.

The most important requirement is that, because the same code is used repeatedly, each variable name is going to be in use several times at once. It is therefore ESSENTIAL that the language can keep the values separate. BBC Basic's LOCAL variables, and procedure and function parameters, make this possible.

A factorial function (e.g. factorial 6 = $6*5*4*3*2*1=720$) can provide a simple example of a recursive routine. Each time that FNFact calls itself, it effectively creates new copies of "n" and "temp", while safely preserving all earlier versions.

```
3000 DEF FNFact(n)
3010 LOCAL temp
3020 IF n=1 THEN temp=1
      ELSE temp=temp*FNFact(n-1)
3030 =temp
```

Of course, that routine could easily have been handled by a loop, as below. What's more, the loop runs noticeably faster than the recursion, as you'll see if you try using them.

```
4000 DEF FNFacta(n)
4010 LOCAL I%,temp
4020 temp=1
4030 FOR I%=1 TO n
4040   temp=temp*I%
4050 NEXT
4060 =temp
```

MORE PRACTICAL RECURSION

Although recursion is rarely essential, there are times where it mimics the way that (some) people approach a problem. A good example is a binary search of a file for a given value.

A binary search (or "chop") starts with a list of items in order (e.g. an alphabetical list of names). To find any specific value, it divides the list in two, picks the middle value and compares it with the value it's looking for. If the middle value is too low, it takes the top half of the list; if it's too high it takes the lower half. It then splits that half into halves and checks the middle value... That's recursion - keep splitting the list in half and looking at the mid-value until you're done.

A good example is searching a telephone directory. (All right, if you're looking for Mr Zygote, you probably wouldn't start in the middle, but you're a human). A conventional routine to do this job was given in the Workshop in the June 1985 issue of Beebug (Vol.4 No.2).

Based on the routine in that issue, here is a recursive function to search the array "data()", between indices "first%" and "last%", for the value "match". It returns either the index of the location holding "match" or, if it cannot find it, the value -1 as an error indicator.

Let's look at this in more detail. Firstly, note that ALL the variables which the function uses are LOCAL, either because they are in the parameter list or because they are explicitly so defined. That's essential, because the values of the variables must be preserved through all the nested recursive calls.

Recursive Search Function

```
10000 DEF FNchop(first%,last%,match)
10010 LOCAL done%,mid%,temp
10020 mid%=(first%+last%)DIV 2
10030 temp=data(mid%)
10040 done%=(temp=match)
      OR ((last%-first%)<2)
10050 IF temp>match AND NOT done%
      THEN mid%=
      FNchop(first%,mid%,match)
      ELSE IF temp<match AND NOT done%
      THEN mid%=
      FNchop(mid%,last%,match)
10060 IF mid%=(last%-1)
      THEN IF (match<>data(mid%))
      THEN mid%=last%
10070 IF mid%>=0
      THEN IF (match<>data(mid%))
      THEN mid%=-1
10080 =mid%
```

Lines 10020-10040 simply find the middle value and check to see if it's the one we were looking for. Line 10050 is the most important line - if no match has been found, it calls FNchop() with revised upper and lower values to force a search of the correct half of the block it was looking at. Once the value is found, or the whole block of data originally sent to the function has been checked fruitlessly, all the nested calls unwind through lines 10060-10080.

The halving function at line 10020 can never actually get to the last point in the array; if the search fails, line 10060 checks the last cell (Mr Zygote?) just in case. Line 10070 sets the error signal if the search fails and, finally, line 10080 sends the index of the searched-for item back to the program which originally called the function.

If all that is a bit complicated and your mind is still boggling gently, try tracing the way through the routine on a sheet of paper. Better still, type the function into your Beeb and experiment with it. Don't forget to set up "data()" first.

LIMITS ON RECURSION

A limitation on recursion is that it can take up a lot of memory. Each recursive call saves all the values so far and, if the nesting goes too far, memory eventually fills up and the program crashes. Try the following program:

→ 49

MAXIMISING YOUR MOUSE

MAX is the latest software product from AMS. Is this another must for every mouse enthusiast? Simon Williams has his doubts.

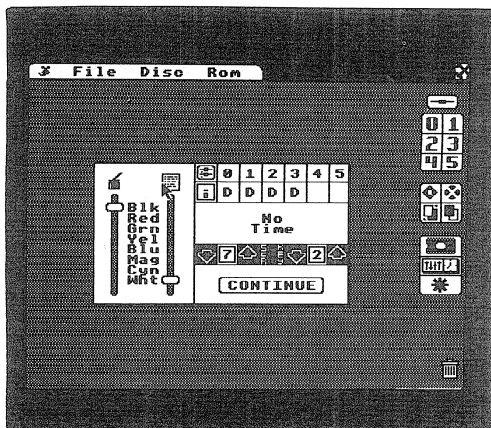
Product: AMX MAX Desktop
Supplier: Advanced Memory Systems Ltd.
 166-170 Wilderspool Causeway,
 Warrington WA4 6QA
 Tel. (0925) 413501

Price: £19.95 for 16k ROM, manual and key strip. (Includes VAT and p&p).

Even really dedicated BBC Micro and Master owners have been seen to look enviously at the Apple Macintosh or IBM PC running GEM or Windows. All these environments set out to make their host micros more 'friendly' and easy to use, via mouse and icon-driven screens. Anyone who has 'clicked' away with a Mac or PC will know that they go a long way to achieving this aim.

There have been several attempts to do the same thing for the BBC Micro, but most have been defeated by a shortage of memory. The latest in this line is MAX, a handy anagram for its producer, AMX. This is a 'front-end on a chip', providing an easy-to-use icon-driven general system manager. MAX allows you to view the contents of discs and ROMs, and to move, delete, copy or run files without ever having to touch a key. You can operate MAX with an AMX mouse or Marconi trackerball, but simple keystrokes will also work for the less wealthy.

MAX fits in any vacant ROM socket in a BBC B, B+, Master or Compact. Fitting is straightforward and well described in the usual high-quality AMX manual. The ROM is called up by typing *MAX, or with f8-Break on a BBC with a DFS, and with f9-Break on a BBC with ADFS, on a Master or Compact.



OPENING A WINDOW

The MAX screen is initially blank, except for a menu bar across the top, and a set of icons down the right-hand side. A setup window opens almost immediately, though, and you can proceed no further until you've clicked your mouse on the 'continue' box. Before you do this though, you may select background and foreground colours by dragging two 'sliders' within a window. You can also tell MAX the number of drives you have connected to your micro, and change the disc filing system, the screen mode and keyboard repeat rate. On a Master, the time is read from the internal clock and is also displayed within the window. You can call the setup window at any time, but unfortunately there's no way of saving your preferred configuration.

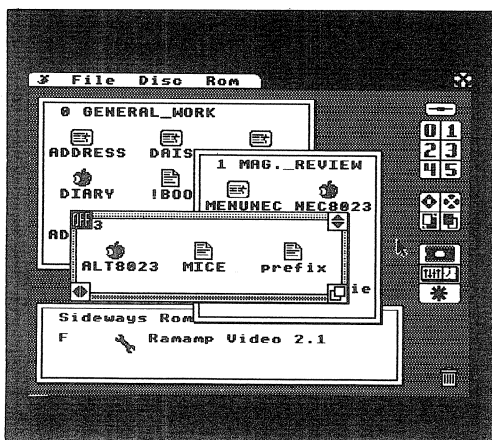
The control icons down the right of the screen are divided into three groups. The top group controls the available drives, and clicking on any of the numbers 0 to 5 will display a directory window for that drive. The second group controls the windows themselves and allows you to expand the selected window to full screen, turn it off, or bring overlapping windows from behind. Some of these facilities are duplicated within the windows themselves, and in use you are likely to stick to those alone. The bottom group of icons call up the ROM display, setup window and star command option. At the bottom right of the screen is a dustbin icon which can be used for deleting files.

When you call up a directory window, it opens to the full size of the screen if there is no other window displayed. If there are other windows on the screen, the new one opens as a small square which may then be moved around the screen or stretched to a different size.

FILE HANDLING

MAX recognises eight different types of file. Files listed in a directory window are normally displayed with a corresponding icon. Some icons are obvious, like the sheet of paper for text files, though others are less so. Files can be sorted alphabetically or by type. You can also scroll the file icons through the window by clicking on the double arrows in the opposite corners of its frame.

You can select a file by clicking on its icon, and double-clicking executes certain types of file. Interestingly, a text file invokes View as its word processor when you execute it. Surely Wordwise (or even Interword) users should equally be catered for!



MAX doesn't act as a 'shell' for your own (or other) programs in the truest sense, as when you run a program you forsake the WIMP environment, and leaving that program takes you back to Basic, not MAX. This is not unreasonable, given that there is nowhere sensible to store the pointers which would recall it. What is poor, though, is that when you copy a file within MAX, the program drops through to mode 7 and then re-enters MAX at the setup window. You have to reset your options and re-open your directory windows before you can copy the next file. Since there is also no Mac or GEM style facility to mark a group of files for copying, it does make the whole process quite long-winded.

The menu bar at the top of the screen offers four pull-down menus, labelled Mouse, File, Disc and Rom. The Mouse menu offers details of MAX itself, and other routes to the setup window and star command function. The File menu allows you to run, delete, rename or lock a selected file, and the Disc menu offers the two directory sorting options. Both File and Disc menus also offer an 'info' option, which provides length and loading addresses for files, and name and drive numbers for discs. The disc information is duplicated in the window itself.

CONCLUSION

The overall impression is that the memory limitations of the BBC Micro are still hampering the implementation of a workable WIMP front-end. MAX does quite well, but seems to work on the basis that providing the same facilities in several guises is a replacement for fitting more genuinely useful features in the first place. The inability of the package to hold a pre-set configuration, group files or remain installed after a simple file copy don't endear it to the busy user. Although competitively priced, MAX doesn't leave enough headroom to be recommended.

```
10 MODE 1
20 N%=1
30 PROCcrash
40 END
50 DEF PROCcrash
60 LOCAL A,B,C,D,E,F,G,H,I,J,K,L
70 PRINT "Iteration No: ";N%
80 N%=N%+1
90 PROCcrash
100 ENDP
```

On my computer it crashes after 46 iterations with a "No Room" message.

In summary, recursion is a bit convoluted, but it's sometimes a very useful trick to have up your programming sleeve. You must be careful to protect all your variables properly, and it can use a lot of memory but, for all that, it has its place and is not just a curiosity.

HT2 130

...>.....*.*.....<
CE-***WATFORD VIEW PRINTER DRIVER-***

Product : ***View Printer Driver ROM***
Supplier : ***Watford Electronics***

***Jessa House,
250 High Str
Watford WD1
Tel. (0923)

Price : £38.95 inc
HT2 160

...>.....*.*.....<
Unlike *DSTK**IT*Wor

T*where
control codes can be embedde
throughout the text. *DSTK**
control of your printer via
so-called *BOLD*'printer dri
to two *BOLD*'highlight' *BC
translated into Escape sequ
priate to your printer. By d
two highlights produce under
emphasizes text. The advanta
approach are that the text i

WATFORD VIEW PRINTER DRIVER

Product : View Printer Driver ROM
Supplier : Watford Electronics

Jessa House,
250 High Street,
Watford WD1 2AN.
Tel. (0923) 37774

Price : £38.95 inc VAT and p & p.

Unlike *Wordwise* and *Wordwise Plus* where
control codes can be embedded anywhere
throughout the text, *View* only allows full
control of your printer via the use of a
so-called 'printer driver'. This allows up
to two 'highlight' markers to be
translated into Escape sequences approp
riate to your printer. By default, the

Although all these features are very
laudable, there is one major disadvantage.
Not only do all the extra forms of high-
lighting distort the otherwise formatted
display in edit mode, but because *View*
treats many of Watford's innovations as
pure text, the resulting printout will
often end up incorrectly formatted. I
found the best approach was to finish
editing and formatting before adding any
extended highlights, but much fiddling
around, with frequent previews (a la
Wordwise) will often be necessary to get
things right.

Finally, if your printer is not in the
list catered for by Watford's ROM, this
versatile piece of software allows you to
create your own version. Printer drivers
created in this way cannot, of course, be
saved in the EPROM, and must be saved to
disc or tape, thereby losing some of the
convenience of Watford's ROM approach.
Furthermore, the Watford ROM must still be
installed in your machine when you reload
a printer driver that you have created.

A simple menu screen can be called up
to create your own printer driver. Often,
this can be achieved most easily by taking
one of the built in drivers as a starting
point, and the 15 drivers available are
fully documented in the manual. When the

new printer driver has been created, there
is an excellent 'test' function which will
send a test sequence to the printer so
that you can check that the new printer
driver is performing correctly.

The supporting 64 page manual is
adequate, but is not always easy reading.
It is somewhat variable in the quality of
its printing, and it can sometimes be
difficult to distinguish between serious
information and examples of the use of
highlights. There is, in addition, one
glaring and crucial error, which could
well undo all the help that Watford has
set out to provide. That is the frequent
use of the underline character instead of
the hyphen to represent one of the
highlight markers. This can only confuse.
The manual also fails to state how the
printer generator program should be
invoked (*PRINTER GENERATOR), a quite
cripping omission.

That said, Watford's new ROM has much
to offer the *View* enthusiast, particularly
those who wish to make full use of the
many features of recent dot matrix
printers (but see our own multi-character
printer driver in this issue!). Whether
the results justify the price you must
decide for yourself.

HINTS HINTS HINTS HINTS HINTS HINTS

and tips and tips and tips and tips and tips

Preserving Tube Text

Owners of a 6502 second processor and Wordwise Plus may be interested to know that any text in Wordwise Plus can be recovered after the machine has been switched off, so long as the tube has not been switched off. This is because all text is stored in the second processor memory, rather than the BBC memory.

Mick Allen

Clearing a REPEAT Loop

The most obvious way to reset all arrays within a running program is to use CLEAR, and then dimension each array again. But you should not do this within a REPEAT loop (or a FOR-NEXT loop) because all loop pointers (and of course FOR-NEXT loop variables) are also cleared. If it is essential to clear arrays within a loop, you could either resort to a GOTO loop (perish the thought!), or clear the array one element at a time using a separate FOR-NEXT loop.

David Graham

FX80 Multi-font Printout

Using ESC 33 you can access almost all of the FX80's different fonts. This allows the use of a short program to print an example sheet showing all 64 different modes (though you may think that there is a certain amount of duplication!). The program is based on one published in the FX80 manual.

```
10 REM FX80 MULTIMODE PRINT
20 VDU 2
30 FOR mode=0 TO 63
40 VDU 1,27,1,33,1,0
50 PRINT"Mode ";mode;
60 VDU 1,9,1,27,1,33,1,mode
70 PRINT"ABCDEF abcdef 1234
567890"
80 NEXT:VDU 3
```

David Graham

Zero Page Workspace

Machine code routines often need some zero page workspace. Contrary to popular belief, locations &70 to &8F are NOT reserved for the user. Locations &00 to &8F are allocated to the current language: Basic allows the user &70 to &8F, but other languages do not (e.g. View uses all the locations from &70 to &8F, and can crash if these are tampered with).

If zero page locations are needed, the original contents should be restored afterwards (the exceptions being &A8 to &AF which are allocated as star command workspace - for both ROMs and library routines stored on the current filing system).

Andrew Benham

Basic BOOTs

When building a !BOOT file, remember that the machine might not be running Basic as the current language. If Basic commands are to be included in the !BOOT file, make the first line '*BASIC'. (The same principle applies to any language-dependent !BOOT file).

Andrew Benham

1770 Free Gift

The sideways RAM commands included in the Acorn 1770 DFS provide a useful undocumented function: a block move which will work across the tube.

The commands *SRREAD and *SRWRITE (documented in the 1770 DFS manual) will work with any I/O processor address in the <sram start> field.

For the SRWRITE operation, any ROM socket can be specified (as a dummy variable), but for the SRREAD operation the socket specified must contain sideways RAM.

The source/destination address uses 32 bit addressing, whereas the sram address uses only 16 bits (but is forced to be in the I/O processor), for example:

```
*SRWRITE 00003000 +20 E00 F
```

will copy &20 bytes from &3000 in the second processor to &E00 in the I/O processor.

Andrew Benham

Double Blurr

A quick way to achieve 'double height text' in mode 1 on a BBC Micro is to use the following line:

```
VDU 240,23,0,0,255,0,0,0,0,0,0,0
```

This re-programs one of the 6845 display registers to produce the blurred effect!

P.Mudhan



POSTBAG



POSTBAG

Mastering Astadd

Those people who have tried to get the popular BEEBUG CAD program ASTAAD (Vol.2 Nos.7 & 9) to work on the Master will have found that the text option does not work properly. This is because the character definitions on the Master are not held in the Operating System, but in the 12K private RAM.

To allow ASTAAD to function properly, change line 510 to read:

```
510 ?&900=
      ASC(MID$(T$,C%,1))
```

and add these new lines:

```
465 OX=X%:OY=Y%
512 X%=0:Y%=9:A%=10:
      CALL &FFF1
514 A%=&901
516 X%=OX:Y%=OY
```

Alan Hatfield

ASTAAD was one of the most popular programs that we have published in BEEBUG, and still rates very highly in comparison with today's much higher standards. We are thus very pleased to publish this information, and maybe give ASTAAD a new lease of life.

Floppywise Master

The review of our (Software Services) Floppywise Master in BEEBUG Vol.5 No.7 contains so many misconceptions and inaccuracies that we feel entitled to put the record straight.

The ROM is not an update of Floppywise - that distinction goes to Floppywise Plus. This latest offering

gives the same facilities to Master owners.

We are criticised for having nothing new in the ROM. Its novelty lies in the machine it is used on. We intended to give people all the facilities they were used to on the Model B which were not available on the Master 128. At the same time each command was upgraded to make use of the new features of the Master.

We were also accused of including two commands already present on the Master. Anyone conversant with this machine is well aware of the limitations of the SRLOAD command - it either takes ages to load, or overwrites your program. Our RAMLOAD command loads instantly with no disruption to work in hand.

The reviewer is entitled to his opinion, but I think some consideration should be given to the facts before expressing it.

G.T.Boden
Software Services

Through lack of space we are unable to reproduce fully the letter from Software Services, but the salient points are as above. Any review is necessarily an expression of one person's views, while lack of space may well mean that features considered important by the supplier are left out. We believe it reasonable to give space to such a letter, but we would rather leave any further judgement to our readers.

Emerging Comma

Jagdish Sah's Part-Merge/Save utility (BEEBUG Vol.5 No.6) does not conform to the general rule that commas are not required after the file name. The addition of:

```
2325 CMP #32:BEQ space
2335 .space
```

makes either of the following formats valid:

```
*MERGE PART,1000,2000
*MERGE PART 1000,2000
      Peter Howard
```

Ah, So

Two years ago I purchased a Canon PW-1080A dot matrix printer which I find excellent. However, I was unable to use the codes to define NLQ characters. I wrote to Canon UK but had no reply. I purchased a book, but this did not include the codes I wanted. I also wrote to the suppliers, but they did not fully answer my question.

Finally, I was lucky enough to be in Japan, and wrote direct to Canon who answered my question simply and fully. Fit a RAM chip (6264) in socket IC3 and turn dip switch 1-5 ON. I can now define NLQ characters without any problem. One point worth mentioning, do not use the printer reset command (OC27,64 in Wordwise) after defining NLQ characters, or they will be erased.

David Else

Other readers may find this information of help.

B

C.P. Yu rounds off his foray into error handling.

MORE COMPLEX ERROR HANDLING

Last month we took a look at the essentials of error handling. In this issue I want to take the subject a little further, going beyond the introductory level. Here we will look at two aspects of error handling: using the Escape key to

perform more complex tasks; and more complex error filtering which will allow a program to respond differently to different kinds of error. But firstly a word of warning.

DESTRUCTION BY ERROR

As I mentioned last month, when an error of any kind occurs in Basic, the loop counters for FOR-NEXT and REPEAT-UNTIL loops are lost. This means that if your program is to recover from an error (including pressing the Escape key), it must restart at some point outside all such loops. But there is a further proviso which must be borne in mind in more complex error handling routines. Unfortunately, if an error occurs at any time when local variables are active, they are made global. To make this clearer, try running program 1.

It is self-documenting, and you will see that if you press the space bar when requested, the procedure will terminate, and the value of A\$ will remain unchanged, but if you press Escape as prompted, while still in the procedure, A\$ will take on the value assigned to it locally during the procedure.

The upshot of this is that you should not use local variables in programs where a full recovery from errors is required, unless of course the restarting point after the error is able to cope with erroneously assigned variables. Another

way around this is to ensure that local variables are only in force at times when the

Escape key is disabled, and when no filing system operations are permitted, since these may generate errors even in a fully debugged program.

Program 1

```
100 REM LOCAL VARIABLE CONFUSION ver 3
110 ON ERROR IF ERR=17 THEN 150 ELSE R
EPORT:PRINT" @ line ";ERR:END
120 A$="RIGHT"
130 PRINT"A$ = ";A$
140 PROClocal
150 PRINT"A$ = ";A$
160 END
170 DEFPROClocal
180 LOCAL A$
190 A$="WRONG"
200 PRINT"Press space to exit procedur
e"
210 PRINT"A$ will remain unaffected"
220 PRINT"But if you press Escape"
230 PRINT"A$ will take on the value as
signed"
240 PRINT"to it locally, and its forme
r value"
250 PRINT"will be lost"
260 IF GET
270 ENDPROC
```

ESCAPING FROM FIELDS

The Escape key provides a very useful way of moving back through the fields of a data input screen. In a typical input screen, data is entered one field at a time, and as each entry is completed the input cursor moves down to the next field. If you make a mistake, pressing Escape would take you to the previous field, and so on. The accompanying program incorporates this idea. It is a little long for a demonstration program, but it has the merit of being fully operational, and although the data entered in each field is not used, it is printed out at the end, as a check that the answers to each question have been correctly picked up by the program.

When you run the program you will see that, for the sake of simplicity, each question only requires a Y/N answer. Keying either, takes you to the next field, and pressing Escape takes you up one field. If you are already at the first when Escape is pressed, you are asked

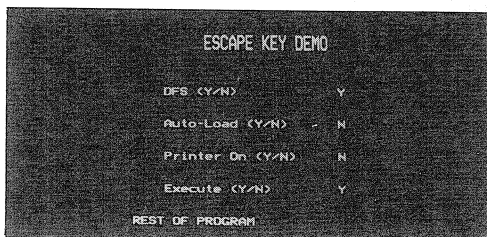

```

10 REM Escape Key Demo v7
20 REM Escape to move up one field
30 REM Return to move down
40 :
100 DIM ans(4)
110 ON ERROR GOTO 1160
120 MODE 7
130 PROCinit
140 ON ERROR GOTO 1160
150 PROCchoice(1)
160 ON ERROR IF ERR=17 GOTO 140 ELSE G
OTO 1160
170 PROCchoice(2)
180 ON ERROR IF ERR=17 GOTO 160 ELSE G
OTO 1160
190 PROCchoice(3)
200 ON ERROR IF ERR=17 GOTO 180 ELSE G
OTO 1160
210 PROCchoice(4)
220 ON ERROR GOTO 1160
230 :
240 REM REST OF PROG
250 PRINT""REST OF PROGRAM"
260 PRINT"Answers are: ";
270 FOR n=1 TO 4:VDU ans(n):NEXT
280 PRINT
290 PRINT"Press any key to repeat ";:I
F GET THEN CLEAR:GOTO 100
300 END
310 :
1000 DEF PROCinit
1010 FOR A=1 TO 2:PRINTTAB(7,1+A)CHR$13
1 CHR$141"ESCAPE KEY DEMO":NEXT
1020 PRINTTAB(4,7)"DFS (Y/N)"
1030 PRINTTAB(4,10)"Auto-Load (Y/N)"
1040 PRINTTAB(4,13)"Printer On (Y/N)"
1050 PRINTTAB(4,16)"Execute (Y/N)"
1060 ENDPROC
1070 :
1080 DEFPROCchoice(n)
1090 *FX15,1
1100 VDU31,25,4+3*n:IF ans(n)<>0 THEN V
DU ans(n),8
1110 REPEAT:A=GET:UNTIL A=78 OR A=89 OR
(A=13 AND ans(n)<>0)
1120 IF A<>13 THEN ans(n)=A:VDU A
1130 ENDPROC
1140 :
1150 REM ERROR HANDLER
1160 MODE 7
1170 IF ERR<>17 THEN REPORT:PRINT" at 1
ine ";ERR:END
1180 SOUND1,-15,150,5:PRINT"EXIT""Are
You Sure (Y/N)?:":IF GET<>89 GOTO 110
1190 PRINT:END

```

point. If you have moved up one or more fields by pressing Escape, you are offered the previously entered answer to the next question, and Return may be pressed to confirm the original answer, or, as before, "Y" or "N" may be used.

If you take a look at the program, you will see that there are two procedures, PROCinit and PROCchoice. The first prints up the input screen, and the second handles the input of data, validating it, and storing it in the array ans(n). We are particularly concerned here with the way in which the error handling is organised, and to see this it is worth glancing at lines 110 to 220. Line 110 sets up the normal error handler, which exits the program unless Escape has been pressed, in which case it asks the user if he is sure that he wants to quit. Lines 140 to 210 handle the effect of the Escape key, ensuring that each time it is pressed, the program moves up one field. The most important thing to note here is that each line pointed to by an ON ERROR statement contains a new ON ERROR statement with a destination a little higher up the program.



MORE COMPLEX ERROR FILTERING

In the above example, a simple form of error filtering was used to distinguish between Escape-key-generated errors and all other sources of error. It is sometimes useful to take the idea of error filtering one stage further. For example, it is often essential that a program should respond differently to a filing system error than to a Basic error. For example if a particular file cannot be found on the current disc, the user should be prompted to change either the disc or the filename that he has given, and the program should, as far as possible, continue from where the error arose. If on the other hand the program hits a syntax error, it would be quite pointless to attempt to continue with the program;

whether you wish to exit the program; and the program is terminated only if "Y" is pressed, Escape being disabled at this

instead the program should be terminated, giving a full error message.

If you take a look at the table of error types and their corresponding error numbers you will see that they have been conveniently grouped. Model B users should refer to section 46 of the User Guide. Master users are less fortunate in that error tables are spread over the two parts of the Reference Manual (pages H.4-2, I.4-2, J.4-4, J.8-5 and M.2-1). As you will see, Basic error numbers lie below 128, while filing system error codes are all above 128. We can thus design an error handler which takes account of this.

Depending on the kind of program which you are writing, there may be other types of error which you wish to filter out for special treatment. One obvious candidate is error number 18, division by zero. You may have a perfectly debugged program, but if the user enters certain parameters as zero, then a division by zero error may occur, and unless you take account of this in the error handler, it will cause the program to crash. Of course, you could check all user input in an attempt to avoid the problem, but this is not always easy. For example, certain "zero" inputs would probably be quite legitimate; and it is only in combination that they may cause a problem. The best way around this is usually to trap the division by zero error, and advise the user accordingly. Of course not all division by zero errors are the result of inappropriate user input, some arise from bad programming, and these should be resolved before the error handler is incorporated.

The bones of a routine to trap filing system and division by zero errors are given in the following example. This routine responds in three different ways according to the error number. If Escape is detected (error 17), the program is resumed at line 200. This allows Escape to take the user to a particular point in the program. As we said earlier, this must not be inside a FOR-NEXT or REPEAT-UNTIL loop of any kind. Note the new ON ERROR statement at line 210. This is necessary in order to cancel the effect of the ON ERROR OFF encountered in line 5000.

If any error other than Escape is detected, then line 5020 is executed to clear the screen and print the error

message. This is required whatever the nature of the error. If the error is not in the program itself (i.e. it is a filing system or division by zero error) then PROCspace is called (line 5030) to produce a warning beep, and leave the error message on the screen until the user presses a key. Note that the error line number is not given since this should be of no consequence. Once a key is pressed, the program resumes at line 120. Note that the error handler leaves the screen in mode 7, so that the main program must reset this to the desired mode as a matter of priority. If on the other hand a non-recoverable Basic error occurs, line 5040 prints the error line number to accompany the error message already printed, and the program goes through an exit routine which resets any required parameters (line 5050).

Even now we could still take this much further, allowing filing system or division by zero errors to take the user not to a common starting point early in the program, but to a point very close to where his error occurred. But this involves a multiplicity of ON ERROR statements, and is beyond the scope of the present article. It is hoped nevertheless that this small foray into error handling will encourage the reader to make better use of errors, and to create tailor made error handling routines for all his major programming projects.

```
100 ON ERROR GOTO 5000
110 PROCinit
120 REM SET MODE HERE
130 REM START OF MAIN PROGRAM LOOP
200 REM GO TO HERE IF ESCAPE PRESSED
210 ON ERROR GOTO 5000
.... etc
4990 REM ERROR HANDLER
5000 ON ERROR OFF
5010 IF ERR=17 THEN GOTO 200
5020 MODE 7:REPORT
5030 IF ERR>128 OR ERR=18 THEN PROCspace:GOTO 120
5040 PRINT" at line ";ERL
5050 *FX4,0 etc
5060 END
5070:
5080 DEFPROCspace
5090 ON ERROR GOTO 5000
5100 PRINT!"Press space to continue"
5110 SOUND1,-15,200,10:IF GET
5120 ENDPROC
```

LABYRINTH

Yet another top-quality game for the Beeb, this time from Tom Price. The graphics are brilliant, the sound effects superb, and the game itself is a real corker. What more can we say?

Poor old Mike, the reckless explorer, has stumbled into a massive maze, packed with nasties and patrolled by man-eating ghosts. Can you help him escape from the wicked grasp of the Labyrinth? Let's hope so, for his sake...

Mike can spin the turnstiles to shut out the pursuing spooks, and may even coax them into the lethal pumpkins, getting rid of both pumpkin and ghost simultaneously. Quite apart from those hazards, however, our luckless hero still has to avoid the lethal poison, and he must gather up all the green gems that are lying on the floor before the door to the next room will open.

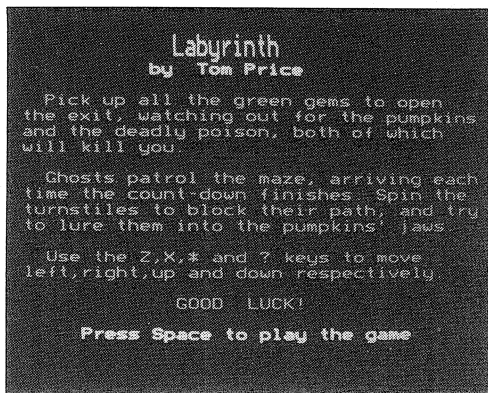
Use the 'Z' and 'X' keys to move Mike left and right, with '*' and '?' moving him up and down. The program, which includes a high score table, is in two separate parts; the first CHAINing the second. Type in part one and save this as 'LABRNTN', then enter part two and save this as 'LAB2', remembering to save it straight after part one if using tape. Be particularly careful when entering the succession of VDU calls in lines 2310 to 2380. Then CHAIN the program 'LABRNTN' to start the program.

Those with a system where PAGE is higher than &E00 should include the following move-down routine at the start of the second program:

```
1 IF PAGE<&E01 THEN 10
2 *K.0 *T.|MFORA%=0TO (TOP-PAGE)STEP4
  :A%!=&E00=A%!PAGE:NEXT |MPAGE=&E00|M
  OLD|MDELET1,4|MRUN|M
3 *FX138,0,128
4 END
```

The maximum number of ghosts is controlled by the variable ng% at line 130 in program two. It is usually set to 1, meaning that no more than two ghosts are allowed at a time, but raising the value within the limits of memory allows a greater number. This, however, will slow the game down. One solution is to lower the value of del% (also line 130), but this has the side effect of creating a noticeable difference of speed between when there are ghosts and when there are not. Experiment with the values as you see fit.

The names for the high score table are held in a DATA statement at line 3470, if you wish to alter them.



```
10 REM Program LABRNTN
20 REM Version B0.1
30 REM Author Tom Price
40 REM Beebug March 1987
50 REM Program subject to copyright
60 :
100 MODE 7
110 RESTORE
120 FOR I%=224 TO 253:VDU 23,I%
130 READ a$:FOR J%=0 TO 7
140 VDU EVAL("&"MID$(a$,J%*2+1,2))
150 NEXT
160 FOR I%=0 TO 21
170 READ I%?&A00,I%?&A20
180 NEXT
190 CHAIN"LAB2"
200 :
1000 DATA 2060202070003828
1010 DATA 7010704070003828
1020 DATA 7010301070003828
1030 DATA 2838001C14141C00
1040 DATA 1E3F3F35353F3B1E
1050 DATA 78FCFCACACFCDC78
```

```

1060 DATA 1E1C3C38387060C0
1070 DATA 78383C1C1CE0E0603
1080 DATA 0000000000000000
1090 DATA 0000000000000000
1100 DATA 001C3E2A6B7F7763
1110 DATA 7F41557F3E3E1C00
1120 DATA 1818003C5A182424
1130 DATA 1C3E2A363E141C00
1140 DATA 2263160C18346322
1150 DATA 1C3E2A3E3E1C081C
1160 DATA 1C1C1C1C7F7F7F7F
1170 DATA 1C1C1C1C1C1C1C1C
1180 DATA 008181EF0042427E
1190 DATA 0024243C00181800
1200 DATA 387C547C7C381038
1210 DATA 3E7F1C1C1C143677
1220 DATA 1C3E2E7E3E1C081C
1230 DATA 7E3F1C3E3EB7E346
1240 DATA 5E7F3D1C3C3C6EE6
1250 DATA 387C747E7C381038
1260 DATA 7EFC387C7CDDC762
1270 DATA 7AFEB383C3C7667
1280 DATA 3F5C1E3F333663E0
1290 DATA FC3A78FCCC6CC607
1300 :
1310 DATA 0,1,0,3,0,9,1,5,1,7
1320 DATA 2,3,2,8,3,1,3,4,3,9
1330 DATA 4,3,5,1,5,5,5,7,5,9
1340 DATA 6,2,6,4,7,6,7,8,8,1
1350 DATA 8,3,8,9

```

```

10 REM Program LAB2
20 REM Version B0.1
30 REM Author Tom Price
40 REM Beebug March 1987
50 REM Program subject to copyright
60 :
100 HIMEM=&2FF0
110 ON ERROR GOTO 290
120 :
130 ng%=1:del%=50
140 PROCinstr
150 PROCinit:PROCstrings
160 REPEAT VDU 22,2:VDU23;10,32;0;0;0
170 PROCvars:PROCscreen
180 REPEAT PROCsetup
190 REPEAT PROCman
200 IF TIME>-1 PROctime
210 FOR H%=0 TO ng%
220 IF Z% H%=ng%:GOTO 240
230 PROCghost(H%)
240 NEXT:UNTIL Z%
250 IF Z%>1 PROCdead ELSE PROCscreen:I
F T%>25 T%=T%-25
260 UNTIL Z%=3:PROChall
270 UNTIL FALSE
280 :
290 ON ERROR OFF
300 MODE 7:REPORT:PRINT" at line ";ERL
310 END

```

```

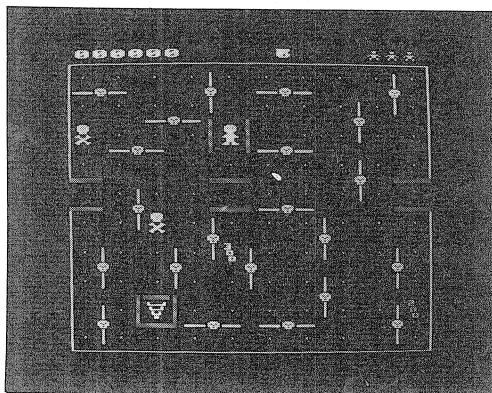
320 :
1000 DEF PROCinit
1010 DIM man$(3,1),ghost$(1),a$(8),x%(n
g%),y%(ng%),on%(ng%),dir$(ng%),ix%(3),iy
%(3),p%(3),tune$(1),hi%(9)
1020 tune$(0)="A8D8H8M8I@H4F4D4C4A8"
1030 tune$(1)="M6M3K3J3H3F6A3F6"
1040 RESTORE
1050 FOR I%=0 TO 3
1060 READ ix%(I%),iy%(I%)
1070 NEXT
1080 FOR I%=0 TO 7
1090 READ $(&900+I%*16)
1100 NEXT
1110 FOR I%=0 TO 7
1120 hi%(I%)=1E4-1E3*I%
1130 NEXT
1140 ENVELOPE 1,1,-1,0,0,-1,-1,-1,40,-6
,-2,-2,80,80
1150 ENVELOPE 2,1,-4,1,0,2,5,4,100,-1,-
1,-2,90,75
1160 ENVELOPE 3,1,1,2,3,10,20,30,110,0,
-2,-3,110,0
1170 ENVELOPE 4,1,1,2,-2,1,1,1,90,-2,-3
,-5,100,75
1180 ENDPROC
1190 :
1200 DEF PROCvars
1210 S%=0:G%=0:T%=150:L%=3
1220 ENDPROC
1230 :
1240 DEF PROCman
1250 A%=0:B%=0:P%=0
1260 C%=Q%:D%=R%:Q%=0:R%=0
1270 IF INKEY(-105) R%=3:Q%=(Y% DIV 3)
MOD 2:B%=3:P%=POINT(X%*64+32,943-Y%*32)
1280 IF INKEY(-73) R%=3:Q%=(Y% DIV 3) M
OD 2:B%=-3:P%=POINT(X%*64+32,1037-Y%*32)
1290 IF INKEY(-98) R%=1:Q%=(X% DIV 2) M
OD 2:A%=-2:P%=POINT(X%*64-32,980-Y%*32)
1300 IF INKEY(-67) R%=2:Q%=(X% DIV 2) M
OD 2:A%=2:P%=POINT(X%*64+96,980-Y%*32)
1310 IF A% B%=0
1320 IF P%=11 OR P%=6 OR P%=9 OR (P%=-1
AND Y%<14) ENDPROC
1330 V%=X%+A%:W%=Y%+B%
1340 IF V%<0 V%=18:Q%=1
1350 IF P%=10 V%=0:Q%=0
1360 K%=&A40+V%DIV2+W%DIV3*10
1370 N%=?K%:IF N%=6 ENDPROC
1380 IF P%>11 PROCbar
1390 IF X%=V% AND Y%=W% AND C%=Q% AND D
%=R% ENDPROC
1400 VDU 31,X%,Y%,32,10,8,32,31,V%,W%,1
7,7:PRINT man$(R%,Q%)
1410 ?K%=0
1420 IF N%=7 Z%=1
1430 IF N%=8 PROCscore(10):E%=E%+1:IF E
%=88 PROCgate
1440 IF N%=4 OR N%=5 Z%=2:a$=a$(N%)

```

```

1450 IF N% AND N%<4 SOUND 3,3,50*N%+25,
1:PROCscore(N%*100)
1460 X%=V%:Y%=W%
1470 ENDPROC
1480 :
1490 DEF PROCbar
1500 SOUND 2,1,40,1
1510 C%=64*(X%-1+A%DIV2+(P%=14)-(P%=15)
)
1520 D%=985-Y%*32-B%*16+48*(P%=12)-(P%
=13))
1530 VDU 29,C%;D%;24,0;64;4;18,3,&8F,
16,24,128;0;184;4;18,3,&8E,16,29,C%+100;
D%-64;24,0;0;1;48;18,3,&8D,16,24,0;88;1;
132;18,3,&8C,16,26,18;&80
1540 ENDPROC
1550 :
1560 DEF PROCgate
1570 PROctune(1)
1580 VDU 24,240;196;328;208;16,26
1590 ENDPROC
1600 :
1610 DEF PROCdead
1620 L%=L%-1:Z%=2-(L%<1)
1630 SOUND &11,0,0,0
1640 FOR I%=220 TO 200 STEP -1
1650 PROCwait
1660 VDU 31,X%,Y%:PRINT man$(R%,Q%)
1670 PROCwait
1680 VDU 31,X%,Y%:PRINT a$
1690 NEXT
1700 IF G%=0 GOTO 1750
1710 FOR I%=0 TO G%-1
1720 VDU 31,x%(I%),y%(I%):PRINT a$(on%
(I%))

```



```

1730 NEXT
1740 FOR I=1 TO 2000:NEXT
1750 ?(&A40+X%DIV2+Y%DIV3*10)=6
1760 VDU 31,X%,Y%:PRINT a$(6)
1770 VDU 31,18-L%;32
1780 ENDPROC

```

```

1790 :
1800 DEF PROCwait
1810 FOR I=1 TO 10:NEXT
1820 SOUND 1,2,I%,1
1830 ENDPROC
1840 :
1850 DEF PROCghost(H%)
1860 IFH%>G% FORI=1TOdel%:NEXT:ENDPROC
1870 IFdir%(H%)<0 VDU31,8,14,17,7:PRINT
ghost$(0):on%(H%)=?&A6C:dir%(H%)=3:ENDPR
OC
1880 I%=x%(H%)*64:J%=y%(H%)*32:p%(0)=PO
INT(I%-32,980-J%):p%(1)=POINT(I%+96,980-
J%):p%(2)=POINT(I%+32,943-J%):p%(3)=POIN
T(I%+32,1037-J%):N%=RND(90):IFN%<57ORp%
(dir%(H%)) dir%(H%)=FNGmove
1890 VDU31,x%(H%),y%(H%):PRINTa$(on%(H%
)):x%(H%)=x%(H%)+ix%(dir%(H%)):y%(H%)=y%
(H%)+iy%(dir%(H%)):VDU31,x%(H%),y%(H%),1
7,7:PRINTghost$(1-dir%(H%)MOD2):on%(H%)=
?(&A40+x%(H%)DIV2+y%(H%)DIV3*10):IFon%(H
%)=5 PROCgdead
1900 IFX%=x%(H%)ANDY%=y%(H%) a$=ghost$(
1-dir%(H%)MOD2):Z%=2:ENDPROC
1910 ENDPROC
1920 :
1930 DEF PROCgdead
1940 ?(&A40+x%(H%)DIV2+y%(H%)DIV3*10)=0
1950 SOUND &11,0,0,0
1960 FOR I%=200 TO 220
1970 PROCwait
1980 VDU 31,x%(H%),y%(H%),17,7:PRINT gh
ost$(I% AND 1)
1990 PROCwait
2000 VDU 31,x%(H%),y%(H%):PRINT a$(5)
2010 NEXT
2020 VDU 31,x%(H%),y%(H%):PRINT a$(0)
2030 G%=G%-1
2040 PROCscore(500)
2050 IF H%=G% ENDPROC
2060 FOR I%=G% TO H%+1
2070 x%(I%-1)=x%(I%):y%(I%-1)=y%(I%):on
%(I%-1)=on%(I%):dir%(I%-1)=dir%(I%)
2080 NEXT
2090 ENDPROC
2100 :
2110 DEF FNGmove
2120 I%=(X%-x%(H%))DIV2:J%=(Y%-y%(H%))D
IV3:A%=(SGN(I%)+1)DIV2:B%=3-(SGN(J%)+1)D
IV2:IFN%MOD9<4ORABS(I%)<ABS(J%) C%=A%:A%
=B%:B%=C%
2130 IFp%(A%)=0 =A% ELSEIFp%(B%)=0 =B%
2140 IFp%(dir%(H%))=0 =dir%(H%)
2150 D%=1:FORI=0TO3:D%=D%-(p%(I%)=0):N
EXT:IFp%(0)=0 D%=D%-1:IFN%MODD%=0 =0
2160 IF p%(1)=0 D%=D%-1:IF N%MODD%=0 =1
2170 IF p%(2)=0 D%=D%-1:IF N%MODD%=0 =2
2180 =3
2190 :
2200 DEF PROCsetup

```

```

2210 Z%=0:G%=0
2220 R%=0:Q%=0
2230 ?&A58=0
2240 X%=8:Y%=8:VDU 31,8,8,17,7:PRINT ma
n$(0,0)
2250 TIME=-T%:U%=11:PROctime
2260 ENDPROC
2270 :
2280 DEF PROCscreen
2290 E%=0
2300 FOR I%=HIMEM TO &2FFF STEP 4: I!%=0
: NEXT
2310 VDU 23;10,32;0;0;0:VDU23;12,5;0;0;
0:VDU23;13,255;0;0;0:VDU19,10,0;0;0:VDU19
,9,0;0;0:VDU19,15,3;0;0:VDU19,14,3;0;0:VDU1
9,13,3;0;0:VDU19,12,3;0;0:VDU19,11,5;0;0:VD
U19,10,0;0;0:VDU26:VDU16:VDU4:VDU17,2
2320 PROCscore(0):VDU 17,2,31,19-L%:PR
INT STRING$(L%,CHR$(236))
2330 VDU 5,18;9,25,4,1248;44;25,5,1248;
1007;18;6,25,4,0;12;25,5,1247;12;25,5,12
47;975;25,5,0;975;25,4,1279;44;25,5,1264
;44;25,5,1264;1007;25,5,1279;1007;26,18;
&8B,24,216;100;352;208;16,24,0;486;96;59
2;16
2340 VDU 24,1112;486;1247;592;16,24,472
;486;608;592;16,24,472;676;608;784;16,24
,1264;516;1279;624;16,18;880,24,1112;500
;1247;576;16,24,472;500;608;576;16,24,48
8;676;592;784;16
2350 VDU 24,0;500;128;576;16,24,232;116
;336;208;16,24,1264;532;1279;608;16,18;8
8B,24,240;196;328;208;16,26,18;10,25,4,1
248;500;25,1,0;76;18;6
2360 FOR I%=0 TO 21
2370 VDU 26,25,4,64+128*I%?&A00;24+96*I
%?&A20;237
2380 IF RND(2)=1 VDU 29,128*I%?&A00;96*I
I%?&A20+8;24,0;0;64;4;18;88F,16,24,128;0
;184;4;18;88E,16 ELSE VDU 29,96+128*I%?&
A00;96*I%?&A20-56;24,0;0;1;48;18;88D,16,
24,0;88;1;132;18;88C,16
2390 NEXT
2400 VDU 26,4
2410 FOR I%=0 TO 99:I%?&A40=8:NEXT
2420 ?&A92=7:?&A58=0:?&A68=0:?&A6C=0:?&
A71=0
2430 FOR I%=1 TO 3
2440 REPEAT:X%=RND(10)-1:Y%=RND(10)-1
2450 UNTIL ?(X%+Y%*10+&A40)=8
2460 ?(X%+Y%*10+&A40)=I%
2470 NEXT
2480 FOR J%=4 TO 5
2490 FOR I%=1 TO 2
2500 REPEAT:X%=RND(10)-1:Y%=RND(10)-1
2510 A%=TRUE:K%=X%+Y%*10+&A40
2520 FOR B%=-4 TO 4
2530 C%=K%+B%+7*(B%<-1)-7*(B%>1)
2540 IF ?C%=5 OR ?C%=4 A%=FALSE

```

```

2550 NEXT
2560 UNTIL NOT (?K%<8 OR K%?10=7 OR K%=&
A4E OR K%=&A62 OR K%=&A40 OR K%=&A49 OR
K%=&A9A OR K%=&AA3) AND A%
2570 ?K%=J%:NEXT,
2580 GCOL 0,&80:FOR I%=0 TO 99
2590 VDU 31,2*(I% MOD 10),3*(I% DIV 10)
+2:PRINT a$(I%?&A40);
2600 NEXT:PROCtune(0)
2610 ENDPROC
2620 :
2630 DEF PROCtune(t%)
2640 FOR I%=1 TO LEN(tune$(t%)) STEP 2
2650 P%=(ASC MID$(tune$(t%),I%)-32)*4-9
6:F%=(ASC(MID$(tune$(t%),I%+1))-48
2660 d%=(ASC(MID$(tune$(t%),I%+1))-48
2670 SOUND 1,t%*2+2,P%,F%
2680 NEXT
2690 REPEAT UNTIL ADVAL(-6)=15
2700 ENDPROC
2710 :
2720 DEF PROChall
2730 FOR I=1 TO 4000:NEXT
2740 CLG
2750 VDU 22,7,23;10,32;0;0;0;0,10
2760 J%=9:FOR I%=7 TO 0 STEP -1
2770 IF S>hi%(I%) J%=I%
2780 NEXT:IF J%>6 THEN 2830
2790 FOR I%=7 TO J%+1 STEP -1
2800 hi%(I%)=hi%(I%-1)
2810 $(&900+I%*16)=$(&8F0+I%*16)
2820 NEXT
2830 hi%(J%)=S%
2840 FOR I%=0 TO 1:VDU &8D81;:PRINT SPC
(12)"Hall of Fame":NEXT
2850 PRINT
2860 FOR I%=0 TO 7
2870 PRINT'SPC(5);I%+1;". ";$(&900+I%*1
6);TAB(24);" ... ";RIGHT$("00000"+STR$(h
i%(I%)),6)
2880 NEXT
2890 *FX 15,1
2900 IF J%<9 PROCcenter
2910 PRINTTAB(6,23) CHR$(83) "Press Spac
e to play again"
2920 REPEAT UNTIL GET=32
2930 ENDPROC
2940 :
2950 DEF PROCcenter
2960 PRINTTAB(12,23) CHR$(85) "Enter you
r name"
2970 VDU 31,8,5+J%*2,23;10,114;0;0;0
2980 PRINT SPC(15) TAB(8,5+J%*2);
2990 !&70=&900+J%*16: !&72=&FF2010
3000 A%=0:X%=&70:Y%=0:CALL &FFF1
3010 VDU 23;10,32;0;0;0
3020 ENDPROC
3030 :
3040 DEF PROCinstr

```

```

3050 VDU 22,7,23;10,32;0;0;0,10
3060 FOR I%=0 TO 1:VDU &8D82;:PRINT SPC
(12)"Labyrinth":NEXT
3070 PRINTTAB(11) CHR$&83"by Tom Price
"
3080 PRINT TAB(0,24) STRING$(20,CHR$&85
+CHR$8+CHR$11)
3090 VDU 28,1,24,39,4
3100 PRINT"" Pick up all the green gem
s to open the exit, watching out for t
he pumpkinsand the deadly poison, both o
f which will kill you."
3110 PRINT"" Ghosts patrol the maze, a
rriving eachtime the count-down finishes
. Spin the turnstiles to block their pat
h, and tryto lure them into the pumpkins
' jaws."
3120 PRINT"" Use the Z,X,* and ? keys
to move left,right,up and down respe
ctively."
3130 PRINT"SPC(13)"GOOD LUCK!"
3140 PRINT"CHR$&83 SPC(4)"Press Space t
o play the game"
3150 *FX 15
3160 REPEAT UNTIL GET=32
3170 ENDPROC
3180 :
3190 DEF PROCscore(I%)
3200 S%=S%+I%
3210 VDU 17,3,30:PRINT RIGHT$("00000"+S
TR$(S%),6)
3220 ENDPROC
3230 :
3240 DEF PROCtime
3250 U%=U%-1:IF U%=0 U%=10:IF G%<2 x%(G
%)=8:y%(G%)=14:on%(G%)=0:dir%(G%)=-1:G%=
G%+1

```

```

3260 VDU 17,7+6*(U%=1),31,10;:PRINT SPC
(- (U%<10));U%=TIME=-T%
3270 ENDPROC
3280 :
3290 DEF PROCstrings
3300 d$=CHR$8+CHR$10:a$(0)=" "+d$+" "
3310 a$(4)=CHR$17+CHR$7+CHR$237+d$+CHR$
238:a$(5)=CHR$17+CHR$1+CHR$234+d$+CHR$23
5:a$(6)=CHR$17+CHR$7+CHR$240+d$+CHR$241
3320 a$(7)=CHR$17+CHR$7+CHR$242+d$+CHR$
243:a$(8)=CHR$17+CHR$2+CHR$232+d$+CHR$23
3
3330 FOR I%=1 TO 3
3340 a$(I%)=CHR$17+CHR$I%+CHR$(223+I%)+
d$+CHR$227
3350 NEXT
3360 FOR I%=0 TO 1
3370 ghost$(I%)=CHR$17+CHR$7+CHR$(I%+22
8)+d$+CHR$(I%+230)
3380 FOR J%=0 TO 3
3390 READ A%,B%
3400 man$(J%,I%)=CHR$17+CHR$7+CHR$A%+d$
+CHR$B%
3410 NEXT,
3420 ENDPROC
3430 :
3440 DATA -2,0,2,0
3450 DATA 0,3,0,-3
3460 :
3470 DATA King Beebug,Frizzz,Ed Skin,Bo
wl,Zits,Slap,Y.Bear,Pot
3480 :
3490 DATA 239,245,246,247,249,250
3500 DATA 239,252,239,245,246,248
3510 DATA 249,251,244,253

```

FILER - THE SORT PROGRAM (Vol.5 No.7)

The last program in the Filer series of programs has no reported errors and needs no further additions.

HELP SYSTEM

The Filer disc implements a help system using the * command *HLP <command>. This accesses a file (H.DATA), an ASCII file (in fact, created with View) with one entry for each command. All Filer programs implement * commands, so this provides an

excellent way of implementing a common help system.

CONCLUSION

The Filer series has proved very successful, and we hope that many members will continue to use and improve this system. It would also be churlish not to thank those many readers who have written in with suggestions or requests for improvements which have contributed to my thinking in this (and earlier) articles.

```

2050 ENDPROC
2060 :
2070 DEFPROCls
2080 MOVE50,30:PRINT"L - Load":MOVE750,
30:PRINT"S - Save"
2090 ENDPROC
2100 :
2110 DEFPROCfilename

```

```

2120 GCOL3,6
2130 MOVE50,30:INPUT"Name:"fn$
2140 MOVE50,30:PRINT"Name:";fn$
2150 ENDPROC
2160 :
2170 DEFPROCoscli(CL$)
2180 $&900=CL$:X%=0:Y%=9:CALL&FFF7
2190 ENDPROC

```


BEEBUG MAGAZINE is produced by BEEBUG 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 Limited.

BEEBUG Publications Ltd (c) 1987

Editorial Address

BEEBUG

Dolphin Place

Holywell Hill

St. Albans

Herts. AL1 1EX

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.

BEEBUG MEMBERSHIP

Send all applications for membership, membership renewals, membership queries and orders for back issues to the address below. 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, B.F.P.O., Channel Islands

£19.00 - Rest of Europe

£23.50 Middle East

£26.00 Americas & Africa.

£28.00 Elsewhere

BACK ISSUES

(Members only)

Volume	Single issues	Volume sets (10 issues)
1	40p	£2.50 (8 issues) – issues 1 and 8 out of stock
2	50p	£3.50
3	70p	£5.50
4	£1	£8.50
5	£1.30	—

Please add the cost of post and packing as shown:

DESTINATION	First issue	Each subsequent issue
UK, BFPO + CH. IS.	40p	20p
Europe + Eire	75p	45p
Elsewhere	£2	85p

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

Back issues are for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

BEEBUG

Dolphin Place

Holywell Hill

St. Albans

Herts. AL1 1EX

Hotline for All non-technical queries
i.e. orders, membership subscription, membership queries etc.

St. Albans (0727) 40303

Manned Mon-Fri 9am-4.30pm

(24hr Answerphone Service for Access/

Visa orders and subscriptions

Technical Queries

St. Albans (0727) 60263

Mon-Fri 10am-4pm

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

Magazine Cassette/Disc

MARCH 1987

CASSETTE/DISC CONTENTS

MULTI-CHARACTER PRINTER DRIVER FOR VIEW — really lets you use all your printer's facilities when printing View text.

VIRTUAL ARRAYS (Part 2) — two more techniques described for storing large arrays, with driver and demo programs in each case.

THE MASTER SERIES VERSATILE AUTO-ALARM SYSTEM — machine code alarm system and Basic manager program.

NEW FONTS FOR YOUR BEEB — routine and demo program to provide 6 new fonts on your Beeb.

LABYRINTH — yet another staggeringly original game, as colourful and as addictive as ever.

BEEBUG WORKSHOP — all the recursive procedures and functions from this month's workshop.

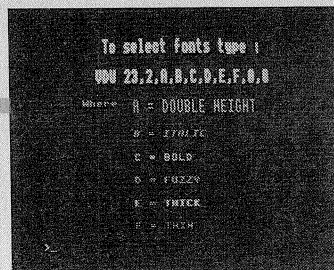
FIRST COURSE — more example programs on the use of error trapping.

GRAPHICS BLOCK MOVE — a most useful and adaptable utility for selectively copying parts of a mode 2 graphics screen.

NEW OS CALLS UNCOVERED — useful example of the new calls

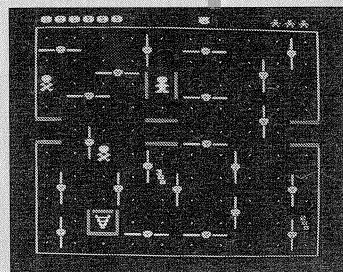
EXTRA FEATURES THIS MONTH

MAGSCAN — data for this issue of BEEBUG (Vol. 5 No. 9).

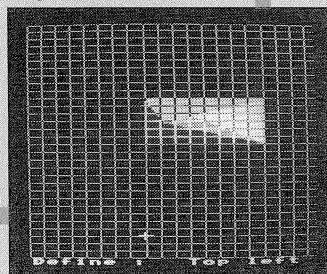


New Fonts

Labyrinth



Graphics Block Move



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

Subscription rates	DISC UK	CASS UK	DISC O'seas	CASS O'seas
6 months (5 issues)	£25.50	£17	£30	£20
12 months (10 issues)	£50	£33	£56	£39

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

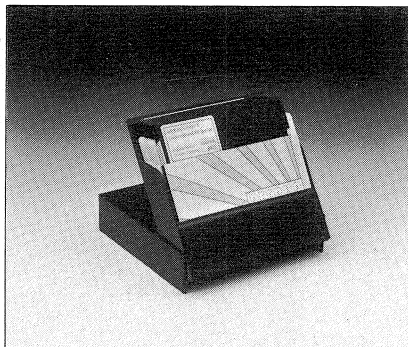
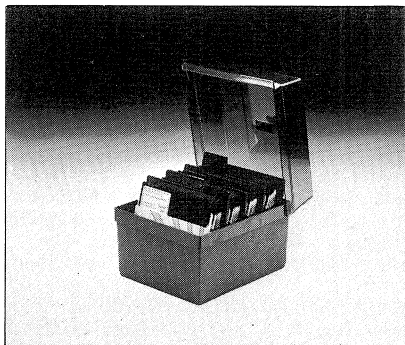
Cassette subscriptions can be commuted to disc subscription on receipt of £1.70 per issue of the subscription left to run.

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

FROM BEEBUG

Beebug members may claim a 5% discount on the price. If you are not yet a member please phone for an application form.

	PRICE	MEMBERS
10 S/S	£9.90	£9.40
25 S/S	£23.00	£21.85
50 S/S	£48.00	£45.60
10 D/S	£10.42	£9.90
25 D/S	£26.47	£25.15
50 D/S	£50.00	£47.50



		PRICE	MEMBERS
10	S/S	£10.42	£9.90
25	S/S	£26.47	£25.15
50	S/S	£50.00	£47.50
10	D/S	£10.90	£10.35
25	D/S	£29.00	£27.55
50	D/S	£53.00	£50.30

- UK Postage 10's — £1.00; 25's & 50's — £3.75



ORDER FORM

Please send me _____ Qty of _____ @ £ _____ (Plus Postage £ _____)

I enclose a cheque for £

☐ Please debit my Access/Visa No.

--	--	--	--

Name: _____ **Membership No:** _____

Address: _____

Post

Code:

NOTE: To claim members prices it is essential to quote membership no. If not yet a member add **£12.90** membership fee, to claim members discount.

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