

Vol.10 No.6 November 1991

BEEBUG

FOR THE
BBC MICRO &
MASTER SERIES

*Acorn
Launch
A5000*



- RUNNING PROGRAMS WITH A JOYSTICK ● STEALTH
- PRINTER SPOOLER UTILITY ● TIMED INTERRUPT ROUTINE

FEATURES

- Patterns from a Keyboard
Epson Compatible or Epson
Incompatible 10
- Running Programs with a Joystick 13
- BEEBUG Workshop:
Simulation Modelling (2) 16
- Printer Spooler Utility 19
- First Course: Program Logic 23
- 512 Forum 29
- Timed Interrupt Routine 34
- Stealth 38
- Wordwise User's Notebook:
Wordwise Plus Teletext Effects (2) 41
- TexBase (3) 47
- 49

REVIEWS

- Acorn Launch A5000 System 6

REGULAR ITEMS

- Editor's Jottings 4
- News 5
- Points Arising 48
- Hints and Tips 57
- RISC User 58
- Postbag 59
- Personal Ads 60
- Subscriptions & Back Issues 62
- Magazine Disc 63

HINTS & TIPS

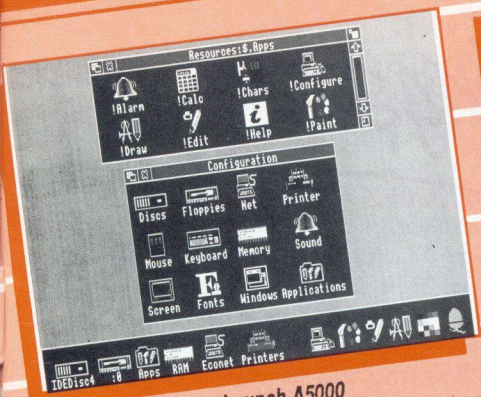
- Extra Help 41
- Hidden ROMs
- Random Assembler
- Assembly Address
- Assembler Square Brackets
- Printer Check
- Listing Variables

PROGRAM INFORMATION

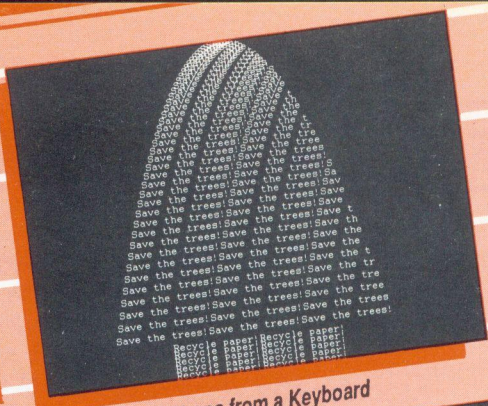
All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the

difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

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

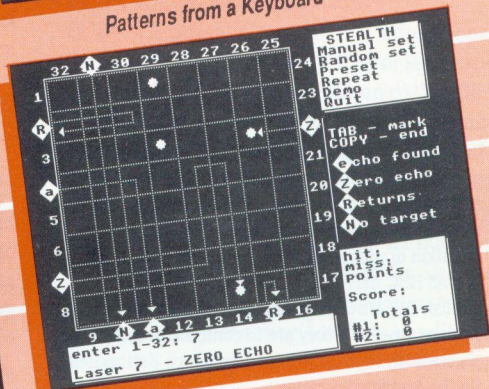


Acorn Launch A5000



Patterns from a Keyboard

- | | |
|-------------------------------|----------------------|
| Printer reset | ESC "@" |
| Pica | ESC "P" |
| Elite | ESC "M" |
| Condensed | CHR\$15 |
| Enlarged | CHR\$14 or ESC "W" n |
| Underline | ESC " " n |
| Italics | ESC "4" |
| Emphasized (Bold) | ESC "E" |
| Double-strike | ESC "G" |
| Superscript, Subscript | ESC "S" n |
| International character sets | ESC "R" |
| Standard line spacing (1/6") | ESC "2" |
| Form length setting | ESC "C" n |
| Skip-over perforation setting | ESC "N" n |
| Left margin setting | ESC "I" n |
| Right margin setting | ESC "O" n |
| Single bit-image graphics | ESC "K" |
| Double bit-image graphics | ESC "L" |

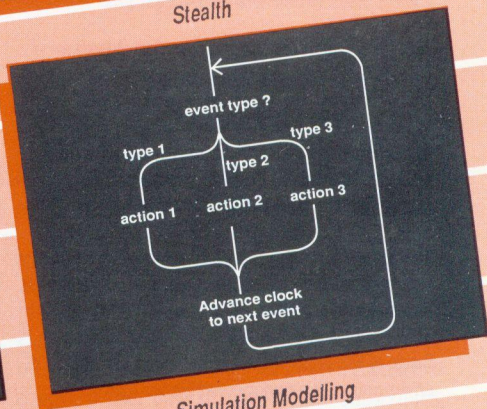


Epson Compatible or Incompatible

| | KEY | -INKEY CODE | FUNCTION |
|---------------|------------------------------|------------------|----------|
| LEFT | Z <Caps Lock> | -98 -65 | FN12 |
| RIGHT | X <Ctrl> | -67 -2 | FN12 |
| DOWN | / (?) . (>) | -105 -104 | FNd2 |
| UP | : (*) ; (+) | -73 -88 | FNu2 |
| FIRE/ JUMP | <Return> <Shift> <Tab> | -74 -1 -97 | FN1 |

Running programs with a Joystick

Stealth



Simulation Modelling

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

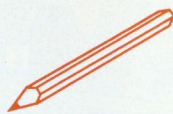


Program needs at least one bank of sideways RAM.



Program is for Master 128 and Compact only.

Editor's Jottings



ACORN'S ARCHIMEDES RANGE

I make no apology for devoting at least part of this editorial to the Archimedes. I know that there are still many BBC micro users who are quite satisfied with the facilities their machine offers and have no intention of upgrading, and it is to support these, and all BBC micro users, that we continue to publish BEEBUG magazine. But all companies have to move forward if they are to continue to compete successfully and survive, and for Acorn that route is via the Archimedes range.

The new A5000, which is reviewed in this issue of BEEBUG, marks a significant advance by Acorn. Not only is it by far the most cost effective model yet in the Archimedes range, but in a number of ways indicates Acorn's intentions of competing in the professional and PC markets which it has until now largely ignored with the Archimedes.

The A5000 is a powerful and high quality computer system; that goes without saying. In appearance, it looks much more like the average PC or Apple Macintosh, and its ability to recognise and use a range of MS-DOS disc formats within the RISC OS environment is a major convenience factor.

This must make the A5000 a much more acceptable product within the traditional PC (and Macintosh) markets. And with a growing range of software, I am sure that this new machine is going to be the most successful Archimedes yet. I also believe it is Acorn's best chance to date of breaking into that lucrative but highly competitive business and professional market, and for one I wish Acorn well.

ACORN USER SHOW

A curious fact of a journalist's life is the not infrequent need to write about future events as though they had already happened. Such is the case with the BBC Acorn User. At the time of writing this is under one week away, yet by the time you read this it will be past history. If all the pre-show hype and rumour is anything to go by, this should have been the most exciting show within the Acorn market for some time. If you attended, you will have had your own opportunity to assess Acorn's new A5000, and also to see much else in the way of new hardware and software.

We will be reporting in future issues on any items of interest to BBC micro users.

BEEBUG MAGAZINE

As I explained last month, the software, hardware and publications departments which existed previously as part of BEEBUG Ltd, have been reorganised as a separate company called RISC Developments Ltd. This is a purely internal move which we all believe will better enable us to expand the range and sales of our own products in the future. Hence RISC Developments Ltd is now formally the publisher of both BEEBUG and RISC User.

The next issue of BEEBUG will be that for December, and this issue, due out late November, will be our main pre-Christmas issue. Look out for special Christmas offers.

ARCHIMEDES PRICE REDUCTIONS

Following the launch of the new Acorn A5000 (see review in this issue), Acorn has announced price changes to its existing model range. The A440 is discontinued and the A420 and A410 have each been reduced by £200 to £1099 and £899 respectively; the R260 UNIX work station and the top-of-the-range A540 are both reduced by £500 to £3495 and £2495 respectively; however the entry level A3000 remains at £599. These prices are quoted ex. VAT.

The A3000 and new A5000 Learning Curve packs now offer optional ink jet printers in the form of the JP150 (an Acorn badged Olivetti model) at an additional £276 (inc. VAT). The A3000 Learning Curve pack with RGB colour monitor costs £999 (inc. VAT), and the A5000 Learning Curve with multisync colour monitor costs £1799 (inc. VAT).

Details are available from all Acorn dealers (including Beebug Ltd), or contact Acorn Computers Ltd., Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN, tel. (0223) 245200.

CORPLAN EXTENDED

Corplan Computer Systems has announced a new utility for users of *Corplan*, the correspondence planner for Wordwise Plus users (see review in BEEBUG Vol.10 No.8). The new facility, *USERBAK*, provides a systematic implementation of backup copies of document discs, and a fully monitored re-saving of Spellmaster user dictionaries from sideways RAM. The backup facility requires the use of twin drives, while dictionary backup requires the prior installation of Computer Concepts' Spellmaster, and sideways RAM which responds to the *SR... commands as on the Master 128 (and Integra B board for the model B). *USERBAK* is supplied on disc for £9.50 post free by Corplan

Computer Systems, Three Gables, 7A Talbots Drive, Maidenhead, Berks SL6 4LZ, tel./fax (0628) 24591.

SOFTWARE FOR SCHOOLS

SCET (Scottish Council for Educational Technology) has announced a total of seven new software packs for schools to run on the BBC model B and/or the Master 128. These include *Fundraiser*, a simulation package for English, *Teletel* for manipulating foreign videotex frames, *Sixth Sense* for technology based problem solving, *Expression*, a graphical environment for outlining and planning text, *En Ville* and *In der Stadt* for modern languages, and a *Basic Program Text Editor*, a utility enabling teachers to edit text embedded in programs, including the provision for other language versions (Gaelic, French, etc.).

All the software has been distributed free of charge throughout Scottish education, but can be purchased from SCET's Information Resource Centre, 74 Victoria Crescent Road, Glasgow G12 9JN, tel. 041-334 3314 from whom further information can be obtained.

MICRONET CEASES OPERATIONS

Micronet, the closed user group for computer enthusiasts operated by Prestel, ceases operations on 31st October. BT, owners of Prestel, claims that despite a user base of some 10,000 subscribers, the level of interest was insufficient to justify continuing the service, and closure will allow BT to concentrate on its commercial activities. Micronet subscribers are being offered the opportunity of transferring to US based CompuServe, but many users of Acorn computers have already found that SID, operated by Acorn, provides a much more relevant service for their needs.



Acorn Launch A5000 System

*Mike Williams assesses a new breed of Archimedes,
the recently announced A5000.*

| | |
|-----------------|---|
| Product | A5000 System |
| Supplier | Acorn Computers Ltd., Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN. Tel. (0223) 245200 |
| Price | A5000 £1499 (ex. VAT) A5000 Learning Curve £1799 (inc.VAT) |

After many months of rumour, Acorn launched the latest addition to the Archimedes range on 27th September, and the new system was the centre of attraction at the recent BBC Acorn User Show. In this issue of BEEBUG we examine the new system, for the benefit of those who might be considering upgrading to an Archimedes, and for those who are just curious to learn a little more of Acorn's latest micro.

Acorn's new A5000 is different in a number of significant ways from previous models in the Archimedes range. First of all, it is badged and marketed as an Acorn machine, not as an Archimedes, though it is still firmly part of the Archimedes family. Secondly, the classic three-box design (of keyboard, computer and monitor) has undergone a major facelift (compared with the previous A300 and A400 series), and has a new and distinctive look to it. Thirdly, and in the long term of greatest importance, the A5000 is the first production computer from Acorn to be supplied with RISC OS 3 (RISC OS being the Archimedes' operating system) as standard.

We shall be looking first at the A5000 as a complete system, and assessing its

likely impact on the Acorn market. We shall also be taking an overview of the major extensions and enhancements to RISC OS. Acorn claims that this version involves over 300 significant improvements compared with the existing version of RISC OS.

THE A5000 SYSTEM

In fact, Acorn has announced two variants of the new system. The first to be released, and available from October, comes with 2 Mbytes of RAM and a 40 Mbyte hard disc. A 1 Mbyte floppy-only system is projected for next year. The current 2 Mbyte version is also supplied complete with an Acorn branded high resolution colour monitor at an all inclusive price, a change from Acorn's previous marketing strategy of offering systems without a bundled monitor. Note, the cut down A5000 announced for 1992 is supplied without a monitor, and it is likely that the two Meg version will similarly become available without monitor in due course.

The keyboard is a standard Archimedes style unit as used with the top level A540 (that is without the awkward plastic keystrip holder), but carrying A5000 and Acorn badges. This uses the familiar and standard PC layout, and is supplied with a 1.5m coiled cable for plugging into the rear of the new system box. The mouse plugs into the rear of the keyboard as before.

The main system unit is 100mm high by 430mm wide by 340mm deep. It is thus wider (from left to right) and shallower (from front to back) than the

Acorn Launch A5000 System

A300, A400 series and the A540. The front fascia is to a completely new, more unobtrusive style than before (which I much prefer), and in this respect the new system differs little from countless other micros (a factor which in a small way may help Acorn's acceptability in the professional and business markets).



The internally fitted 3.5" IDE hard disc (the latest type of disc technology already widely used in the PC market) is mounted to the left (when viewed from the front), with a single indicator LED. Provision is made to the right for up to two floppy disc drives mounted vertically one above the other, and one is supplied as standard. The main on-off switch is just to the right of the floppy drive position, and now, conveniently, controls both micro and monitor.

Looking at the rear of the new machine the power supply (rated 70 W continuous) is to the left, with various

interface sockets (Printer, Serial, Econet, VGA, Stereo Sound, and Keyboard) arranged along the lower edge. Above these, there are four standard-width expansion slots (which will alternatively take two double-width cards) for fitting Acorn and third party expansion cards. There is also a follow through mains socket for powering the monitor. Provision also exists for a future option which would allow a user to connect a PC keyboard in place of Acorn's own design.

Internally, the A5000 boasts an ARM3 processor running at 25 MHz sitting on a completely new design of motherboard. This accommodates 2 Mbytes of RAM which can be expanded to a total of 4 Mbytes. There is also on-board provision to retro-fit the forthcoming floating point accelerator chip (first announced for the A540), which is scheduled for release in the first quarter of next year.

The multisync monitor will support both the 16 and 256 colour VGA standard at a resolution of 640 by 480 pixels.

The whole system is visually very appealing, and with built-in ARM3 processor it is also very powerful. Raw performance, as might be expected, is virtually the same as that of top-of-the-range A540, and some three times faster than the current A400 series. It compares more than favourably with 386 and 486 based PC alternatives.

RISC OS 3

The A5000 is supplied with the latest version of Acorn's RISC OS operating system. Compared with version 2, RISC OS 3 has been enhanced and extended in many significant ways. All the major applications (like Edit and Draw) are

Acorn Launch A5000 System

now in ROM for fast, immediate access (see table 1). This also means that they require much less user RAM when operational. The applications themselves have been improved and extended, particularly Paint, Draw and Edit.

| | |
|-----------|-------------------------|
| Paint | Pixel editor |
| Edit | Text editor |
| Draw | Object-based drawing |
| Alarm | Alarm clock |
| Calc | Desktop calculator |
| Configure | Machine configuration |
| Help | Interactive help system |

Table 1. RISC OS 3 ROM-based applications

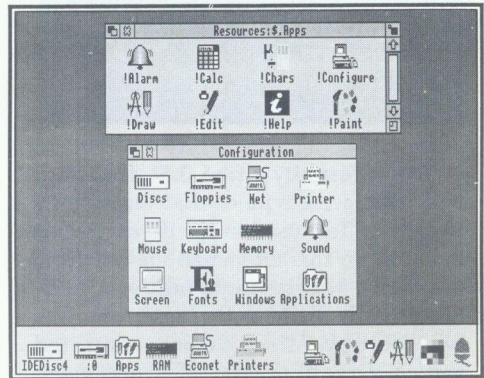
Edit now has a Print option, to be consistent with Draw and Paint; the search and replace facility has become much more powerful (and flexible); there is a menu option to change filetype; and options covering wordwrap and overwrite. Edit will now also accept Basic files, automatically converting them to text for editing, and resaving in tokenised Basic format.

Draw has been significantly improved with regard to text handling (partly via the new outline font manager). Text can now be edited, and text objects manipulated by applying transformations. Paint has a new initial dialogue box to make life easier for the user, and the colour and tools menus appear automatically.

The Alarm application has also been significantly improved, and can also be used to initiate the running of applications at pre-determined times, simply by dragging the application into the Alarm window. There is also a completely rewritten Configure application, which makes configuring a system much easier than before.

The Filer (the part of the operating system which deals with all file operations) been made much more multi-tasking, so that functions such as Copy, Delete, Move, Format and Verify can all be carried out in the background while other tasks are being performed.

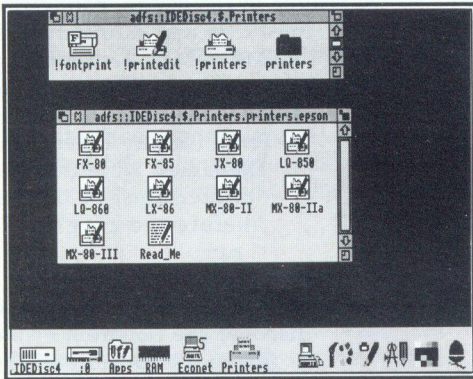
The ADFS has also been extended to support the internally fitted IDE hard disc drive, and with a new disc controller chip it now allows 1.6 Mbytes of data to be stored on high density floppy discs. It also recognises DOS format 360 Kbyte, 720 Kbyte, 1.2 Mbyte and 1.44 Mbyte discs, and Atari 360 Kbyte and 720 Kbyte disc formats. Place one of these discs in the drive, and a window can be displayed (as for an ADFS format disc), showing the root directory of the disc. DOS files can be dragged and accessed as for other files, (though the PC emulator is, of course, needed if you want to run DOS applications).



The ROM-Based resources and new Configuration window

A revised version of the outline font manager is now provided as standard in ROM, together with 12 font styles, and new font families can be readily incorporated into the system. Using the built-in fonts, memory requirements for

font caching have been considerably reduced, releasing memory for other applications. There is also a new application called *FontPrint*, which not only allows the user to relate the A5000's fonts to those used by any PostScript printer, but also permits any non-standard outline font to be downloaded to a PostScript printer, thus overcoming one of the major current limitations regarding the printing of outline fonts to such printers.



Setting up a printer driver showing current Epson support

The printer driver system has also been completely rewritten, with a major part of the new system again in ROM. Selection and customisation of printer drivers has been made much easier, and more than one printer driver can be installed at the same time. This is not only useful for networked systems, where more than one printer may be accessible, but also permits logical printer drivers to be targeted to the same printer, allowing a choice between, say, draft and letter quality modes as required. Furthermore, each printer driver is fully queued, with comprehensive user control, and all printing effectively takes place in the background while the user continues with other tasks.

Lastly, in this roundup of features, the capabilities of the Desktop environment have been extended even further, yet control and use of the system is even easier than before: there are more options and far more of these can be controlled from the Desktop. The screen has also become a sticky backdrop, called a *Pinboard* by Acorn, permitting individual files and directories to be 'stuck' to the backdrop where they remain even when the parent directory viewer disappears.

Directory viewers can be 'tokenised' into a miniature sprite, and also stuck to the Pinboard.

There is also a boot facility so that by using a simple Save box the details of the current Desktop display, including whatever applications are installed on the icon bar, can be saved, and the system rebooted to this state when next switched on.

SUMMARY

Let me say at once, that I am full of admiration for the A5000, and cannot wait to get my hands on one on a permanent basis. The system itself is fast and powerful; the inclusion of a high quality monitor ensures that screen displays are to a professional standard which does full justice to any software; and the advent of RISC OS 3 provides a wealth of features which are genuinely easy to use and to control. At its price, it has to be a runaway winner.

What of existing machines? We understand that the present A440 will be discontinued, and that there will be price reductions on the A410 and A420, but I cannot personally see these continuing for long. Within 12 months I would expect to see these completely replaced by an expanded A5000 series.

Continued on page 22

Patterns from a Keyboard (1)

Jeff Gorman shows how you can produce a huge variety of patterns from the keyboard using your printer.

INTRODUCTION

This application started life as a dodge to knock up a background quickly for the cover of a short-run A4 booklet to be produced on a photocopier. It used the ploy of entirely covering a sheet of paper with a phrase such as a title or slogan. The actual title was printed over the background but it could have been printed on a separate piece of paper (perhaps with a border) and pasted down. Experience shows that there is no need to be limited to meaningful text - it is amazing what effects can be produced by repeatedly printing some of the punctuation marks and other keyboard symbols.

The version of the program given here utilises certain printer controls to elaborate on the original idea by varying the typeface, by printing areas of defined size and of rectangular, triangular or somewhat parabolic outline, and by varying the line spacing. It can now serve as a means of enhancing greeting cards, notices, club literature, overhead projector transparencies and the like. The illustrations accompanying this article will give some idea of the scope.

The various options should work with most printers which accept Epson

codes (see article in this issue of BEEBUG on Epson compatibility), but it might be found, for example, that certain printers only respond to instructions to centre or right-justify text in Near Letter Quality (NLQ) mode.

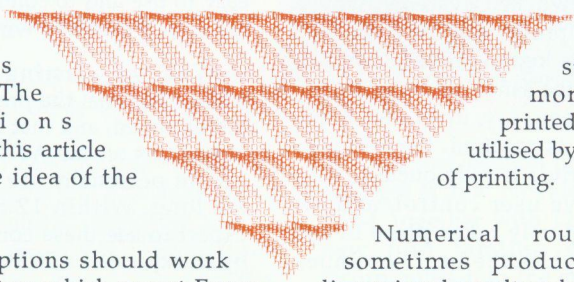
USING THE PROGRAM

Type in the listing, noting that lines 2360 and 2370 start with REMs which temporarily block those options to be added next month. Save as *Ptrn1* (or any other legal filename). Note that the printer must be on line before the program is run.

The best way to discover the potential of the program is to follow the menus and systematically experiment with each of the options, and with each of its modes. Next month's article will indicate something of the workings of the program, but a few preliminary notes follow.

A *graphic unit* can be either a single keyboard character, or a string up to 238 characters long, but a string taking up more than one printed line can only be utilised by the *Joined* mode of printing.

Numerical rounding errors sometimes produce unexpected dimensional results when expanded or



compressed modes are selected, so experiment to establish the exact size of the output before attempting final copy.

The units given for *Feed* are seventy-seconds of an inch (1/72"). A feed of less than six produces overprinted characters. This is probably most useful when making patterns from keyboard characters, somewhat resembling printers tints.

Experimenting with the vertical bar "|" as a graphic unit reveals small lateral inaccuracies when the printer prints on both the outward and inward passes of the head (bi-lateral printing). Uni-lateral printing is also possible, and since there can be subtle pattern differences between the two printing methods, the program offers a choice.

In the final version (next issue) there will be an option to reverse feed the paper back to the starting point, (providing the printer can do so), thereby expanding the scope for design by enabling the juxtapositioning of pattern blocks or printing one tint over another.

It is all very well experimenting, providing one can remember what one has done, so next month's article will also offer an option which prints a report after each run. For now, why not give it a try?

```

10 REM Program Pattern1
20 REM Version B1.1
30 REM Author Jeff Gorman
40 REM BEEBUG November 1991
50 REM Program subject to copyright
60 :
100 ON ERROR GOTO 160
110 DIM typ$(12):MODE 7
120 PROCinit:PROCstyleMenu
130 PROCsetStyle:PROCunits
    
```

```

140 PROCnextStage:END
150 :
160 ON ERROR OFF:*FX3,0
170 VDU2,1,27,1,64,3,26
180 PROCcursor(1):*FX4,0
190 MODE3:REPORT:PRINT " at line "ERL:
END
200 :
1000 DEF PROCcursor(d%)
1010 VDU23,1,d%,0;0;0;0;:ENDPROC
1020 :
1030 DEF PROCinit:s$="*****":*KEY4 RUN|M
1040 no$="1234567890":p=10:depth=0
1050 ov$="N/a":gbu$="N/a":ud$="N/a"
1060 js$="N/a":id$="N/a":feed%=1:bdn%=1
1070 back%=0:spac%=0:REM m%=0
1080 VDU2,1,27,1,64,3,26,1,19
1090 c$=CHR$(134):y$=CHR$(131)
1100 r$=CHR$(129):g$=CHR$(130)
1110 rect=TRUE :unif=TRUE :vert=FALSE
1120 join=FALSE:band=FALSE:crpd=FALSE
1130 left=FALSE:oblq=FALSE:down=FALSE
1140 decr=FALSE:incr=FALSE:text=FALSE
1150 revr=FALSE:unMo=FALSE:end =FALSE
1160 up =FALSE:ENDPROC
1170 :
1180 DEF PROCstyleMenu
1190 PROChead:PROCwnd(0,21,39,3,-1)
1200 RESTORE 1380:FOR a%=1 TO 13
1210 READ col%,option$
1220 PRINT TAB(2) CHR$col% option$
1230 NEXT:PRINT c$ STRING$(38,"*")
1240 PROCwnd(0,24,39,17,-1)
1250 PRINT y$ "Use cursor keys to move
the marker "*****"
1260 PRINT c$ "Key Return to select"
1270 PROCwnd(0,21,39,3,0):ENDPROC
1280 :
1290 DEF PROChead:PROCwnd(0,2,39,0,-1)
1300 PRINT c$ STRING$(38,"*")
1310 PRINT y$ STRING$(7," ") "Patterns f
rom a Keyboard"
1320 PRINT c$ STRING$(38,"*");:ENDPROC
1330 :
1340 DEF PROCwnd(l%,b%,r%,t%,cls)
1350 VDU28,l%,b%,r%,t%:IF cls CLS
1360 ENDPROC
1370 :
    
```

Patterns from a Keyboard

```
1380 DATA 131,"Pica font"
1390 DATA 134,"Elite font"
1400 DATA 131,"Near Letter Quality Mode"
"
1410 DATA 134,"Italic face"
1420 DATA 131,"Subscript Mode"
1430 DATA 134,"Superscript Mode"
1440 DATA 131,"Expanded Mode"
1450 DATA 134,"Condensed Mode"
1460 DATA 131,"Underlined Mode"
1470 DATA 134,"Double Strike Mode"
1480 DATA 131,"Emphasised Mode"
1490 DATA 134,"Own option - 1"
1500 DATA 130,"Finish Selection"
1510 :
1520 DEF PROCsetStyle:y%=1:k=1:num%=0
1530 PROCwnd(0,21,39,3,0):REPEAT
1540 num%=num%+1:k=FNmenu(0,14)
1550 IF k=1 PROCv(64,0,10,"Pica")
1560 IF k=2 PROCv(77,0,12,"Elite")
1570 IF k=3 PROCv(120,1,10,"NLQ")
1580 IF k=4 PROCv(52,0,p,"Italic")
1590 IF k=5 PROCv(83,1,p,"Subscript")
1600 IF k=6 PROCv(83,0,p,"Superscript")
1610 IF k=7 PROCv(87,1,p*0.5,"Expanded"
)
1620 IF k=8 PROCv(15,0,p*1.7,"Condensed
")
1630 IF k=9 PROCv(45,1,p,"Underlined")
1640 IF k=10 PROCv(71,0,p,"Double Strik
e")
1650 IF k=11 PROCv(69,0,p,"Emphasised")
1660 IF k=12 PROCv(0,0,p,"Own option")
1670 UNTIL k=13:VDU3
1680 PROCwnd(0,24,39,17,-1)
1690 PRINT y$ SPC(2)"OK (Y/N) ? ";
1700 IF FNip(1,-1,"YN")="N" VDU2,1,27,1
,64,3:p=10:PROCstyleMenu:PROCsetStyle
1710 PROCwnd(0,20,39,3,-1)
1720 PROCwnd(0,24,39,18,-1)
1730 pitch=p:ENDPROC
1740 :
1750 DEF FNmenu(first%,last%):*FX 4,1
1760 PROCcursor(0):REPEAT
1770 PRINT TAB(0,y%-1)y$***;:k%=GET
1780 IF k%=138 y%=y%+1:PRINT TAB(0,y%-2
)c$ " "
1790 IF y%=last% y%=first%+1
```

```
1800 IF k%=139 y%=y%-1:PRINT TAB(0,y%)c
$" "
1810 IF y%=first% y%=last%-1
1820 UNTIL k%=13:PRINT TAB(2,y%-1)r$
1830 *FX 4,0
1840 *FX12,0
1850 PROCcursor(1):=y%
1860 :
1870 DEF PROCv(c%,e%,a,type$)
1880 VDU2,1,27,1,c%,1,e%,3
1890 p=a:typ$(num%)=type$:ENDPROC
1900 :
1910 DEF FNip(l%,caps,ok$)
1920 PROCcursor(1):a$="":*FX21,0
1930 PRINT STRING$(1%, ".");STRING$(1%,C
HR$(8));:REPEAT asc%=GET
1940 IF asc%=13 AND a$<>"" UNTIL TRUE:=
a$:PROCcursor(0)
1950 IF asc%=127 AND a$="" UNTIL FALSE
1960 IF asc%=127 a$=LEFT$(a$,LEN(a$)-1)
:VDU127,46,8:UNTIL FALSE
1970 IF caps AND asc% >57 asc%=(asc% AN
D 223)
1980 IF LEN(a$)=1% OR INSTR(ok$,CHR$asc
%)=0 VDU7:UNTIL FALSE
1990 PRINT CHR$asc%,:a$=a$+CHR$asc%
2000 UNTIL FALSE
2010 :
2020 DEF PROCunits:PROCwnd(0,20,39,3,0)
2030 PRINT TAB(11,6)c$ STRING$(27,"**")
2040 PRINT TAB(0,7)c$ STRING$(12,"**")
2050 PROCcursor(1):PROCwnd(0,24,38,16,-
1)
2060 PRINT"Enter ""Graphic Unit""
2070 PRINT"(Characters, symbols or text
)"
2080 PRINT"Maximum of 231 characters"
2090 PROCwnd(1,11,38,3,0):INPUT LINE ""
A$:IF LEN(A$) >1 text=TRUE
2100 IF A$="" VDU7:CLS:PROCunits
2110 PROCwnd(0,24,39,16,-1)
2120 PRINT SPC(1)"OK (Y/N) ? ";
2130 IF FNip(1,-1,"YN")="N" PROCwnd(0,2
0,39,3,-1):PROCunits
2140 PROCwnd(0,24,38,17,-1)
2150 PRINT " Graphic unit is:- "; """"
A$ + """"
```

Continue on page 53

Epson Compatible or Epson Incompatible?

Alan Wrigley looks at the thorny question of compatibility between printers.

We've all heard the phrase "Epson compatible" whenever printers, or software which addresses printers, is mentioned. But what exactly does it mean?

In the early days of personal computing, Epson established itself as the dominant manufacturer of low-cost, high-quality printers. The MX-80 was perhaps the first such printer to be sold in large quantities to users of microcomputers, and this was followed by the RX-80 and FX-80. These models proved to be extremely popular.

The growth in the printer market made it possible for many more people to consider purchasing one, and as a result there was a corresponding growth of interest in software applications which could make use of a printer - not just the obvious ones like word processing, but also spreadsheets, databases with report facilities, and a whole host of other programs which could dump information to a printer. In order to work correctly, these programs had to make some assumptions about the printer in use - how to reset it, how to turn on bold, how to set the margins and so on. Since Epson was the major manufacturer, the codes used by their printers became an unofficial standard. Other manufacturers soon realised that if their printers were to sell in large numbers, they would have to recognise the same codes in order to work with the majority of software, and so the concept of Epson compatibility was born.

Unfortunately for the user, this was just the start of the problems. Printer

technology does not stand still, and within the span of models catering for this section of the market there is great variety in the range and complexity of features provided. So although it might be true that certain features, and the codes to access them, are common to most compatibles, there are still a great many pitfalls, as any software writer knows to his cost.

Before we consider compatibility in detail, it is worth noting that some attempts *have* been made by software writers to get around the problem, by providing "printer drivers" which can be tailored to a specific printer, whether Epson compatible or not. Acorn's *View* word processor is a good example of this. The application itself knows nothing about printer codes, relying on the printer driver to translate *View's* own internal codes for bold etc. into the codes needed by the printer. On the Archimedes, this concept has been expanded to the point where it is the preferred method for *all* applications to send output to a printer. This concept fits in well with the multi-tasking environment on the Arc.

WHAT IS COMPATIBILITY?

To start with, what does "Epson compatible" actually mean? Epson has produced a large number of printers over the years, so which model is the standard for compatibility? The programs we publish in BEEBUG magazine and on the magazine disc are tested on an FX-80, and unless stated otherwise this is the model for which compatibility is guaranteed. It is a

Epson Compatible or Epson Incompatible?

sensible standard to follow, since it is (almost) the lowest common denominator. However, the FX-80 is an old printer, and lacks many of the features of more recent models. Later Epsons, such as the LQ series, have a 24-pin print head instead of the 9 pins of the FX, and have many additional features such as NLQ, different fonts, double-height characters etc. So immediately there is a problem - if a program outputs to an Epson compatible printer, does it mean FX or LQ? The difference can be quite significant, as I will explain shortly. And this is just for Epson printers themselves; every time another manufacturer adds a new feature to a printer there is the possibility of incompatibility, particularly if the feature does not already exist on an Epson model.

Facilities you *can* generally rely on are listed in table 1. You can normally expect that any so-called compatible printer will use the Epson codes to access these features, and so software which uses only those functions should work correctly (refer to your printer manual for more details).

Beyond this simple list lies a minefield. For example, staying with 9-pin for the moment, most printers these days have NLQ (near letter quality). In nearly all cases, these use the Epson codes (the LX-80 is the standard here), except for older Canon and Taxan printers which have their own codes. Most printers have a quad-density bit-image graphics mode in addition to the single and double, but a

few budget printers do not. The FX-80 has the capability for downloading your own character definitions to the printer; some other printers do not have this, some follow the FX-80 standard, while others have the facility but use different codes, or even a different character matrix.

| | |
|-------------------------------|----------------------|
| Printer reset | ESC "@" |
| Pica | ESC "P" |
| Elite | ESC "M" |
| Condensed | CHR\$15 |
| Enlarged | CHR\$14 or ESC "W" n |
| Underline | ESC "." n |
| Italics | ESC "4" |
| Emphasized (Bold) | ESC "E" |
| Double-strike | ESC "G" |
| Superscript, Subscript | ESC "S" n |
| International character sets | ESC "R" |
| Standard line spacing (1/6") | ESC "2" |
| Form length setting | ESC "C" n |
| Skip-over perforation setting | ESC "N" n |
| Left margin setting | ESC "I" n |
| Right margin setting | ESC "O" n |
| Single bit-image graphics | ESC "K" |
| Double bit-image graphics | ESC "L" |
| Incremental paper feed | ESC "J" n |

Table 1. FX-80 compatible functions with codes

24-PIN PRINTERS

When we move on to 24-pin printers, a whole new can of worms opens up. For a start, the pins are spaced at a distance of 1/60" rather than the 1/72" of a 9-pin head. This means that, although the codes to set line spacing are identical to the FX-80, the actual vertical movement of the head for a given setting will be incorrect by a factor of 6:5 if the software assumes that a 9-pin printer is connected. Note that this will not affect normal text printing, where a standard line spacing of 1/6" is used in both cases, but it will be noticeable when printing screen dumps, for example, and other instances where the line spacing is varied from the standard.

Epson Compatible or Epson Incompatible?

Manufacturers' top-of-the-range models are invariably 24-pin, so it would be reasonable to expect that most new models will contain features not present on earlier or lower-priced models. If Epson has not managed to get there first, and several other manufacturers develop features simultaneously, you have a recipe for incompatibility, and this is exactly what has happened in some cases. Take double-height characters for example. To my knowledge there are at least 3 standards in existence, and to make matters worse, some printers reverse the carriage before printing in double-height, others do not. This means that even if you can substitute the codes, you still cannot achieve the desired result with a printer using a different standard.

In addition, of course, some printers will have features which are just not present on many other models.

Wide ranges of fonts, shadow and outline printing, extra pitches such as 15 and 20 cpi, triple-width characters and even simulated 48-pin printing using two passes of the print head are just some examples. There are, however, a few functions on 24-pin printers which are fairly common and conform to the Epson standard. These are listed in table 2, and you will find in most cases that the codes for these are adhered to by other printers. These are of course in addition to those listed in table 1, which are all present on 24-pin printers.

There are many other functions with different standards which I have not mentioned, such as IBM character sets

(a whole subject in itself), individual tab settings, 360x360 dpi graphics (24-pin only), sheet feeder selection, absolute print head positioning and so on. If you have ever wondered why software authors cannot get it right with your particular printer, perhaps this article will have given you some idea of the complexities involved. Don't forget, too, that as soon as a piece of software is written which makes use of printer facilities, along will come another printer with unsupported features.

In case the tone of this article seems too negative, I must stress that *most* software will cope quite happily with *most* of the usual functions. It is only if you want to use some of the more advanced or esoteric facilities, such as I have described above, that you will run into trouble. To release the full potential of your printer,

you may well need a good knowledge of programming, but if all you want is a printer which works with your word processor and prints in bold, italic, underline

etc., then you should have no worries whatever Epson compatible you buy.

If you are writing software, particularly for publication either commercially or in the pages of a magazine such as BEEBUG, it is good practice to ensure that it uses only the facilities listed in the tables above, if necessary providing separate modes of operation for 9-pin and 24-pin. If you do this, you won't go far wrong, and there is less likelihood of complaints from disgruntled users later.

| | |
|-------------------------------------|-----------|
| Near letter quality | ESC "x" n |
| Proportional spacing | ESC "p" n |
| Justification | ESC "a" n |
| Reverse incremental paper feed | ESC "j" n |
| Colour selection (where applicable) | ESC "r" n |
| Quad-density bit-image graphics | ESC "Z" |

Table 2. Additional LQ compatible functions with codes

Running Programs with a Joystick

by Simon Myers

Joysticks are popular with users, and many programs published in BEEBUG (and not just games) could potentially be used with joysticks. This article presents a way to run programs with the joystick that can easily be incorporated into many Basic programs with very little knowledge of programming being needed.

INTRODUCTION

Accompanying this article are two sets of functions which should be typed in as *EXEC files. The simplest way to do this is to type them straight in to a text editor such as the Master's Edit or Wordwise - this makes correcting mistakes easier than by using the *BUILD method. Save the listings with names JStk1 and JStk2 respectively. When typing them in be careful not to confuse brackets () with comparison signs < >.

Calls to these functions can then be used in programs instead of tests on BBC Basic's functions GET and INKEY, and the functions themselves check for both joystick and keyboard input. This method is simpler than writing a menu option to select or deselect the joystick, setting a flag variable, and reading this variable each time that input is required.

Two different sets of functions are needed because of the two different methods of reading the keyboard - directly or via the keyboard buffer. The differences between these were explained in *First Course* by David Graham in Vol.5 Nos.4-5, and there is little point in it being repeated here. Except for their highly condensed format so that they can be added even to programs with very little memory left, the functions should not be difficult to follow. Read Mike Williams' *First Course*

article in Vol.5 No.3 for an explanation of the ADVAL keyword.

USING THE FUNCTIONS

JStk1 will be most useful in WIMP (windows, icons, mouse and pointer - or maybe it should be WIJP?) applications and utilities where some sort of cursor is moved to point at different objects which can then be selected. Typically the code to do this looks something like this:

```
100 REPEAT
110 key%=GET
120 IF key%=136 THEN PROCmoveX(-1)
130 IF key%=137 THEN PROCmoveX(1)
140 IF key%=138 THEN PROCmoveY(1)
150 IF key%=139 THEN PROCmoveY(-1)
160 UNTIL key%=13
```

The above lines could form part of a program which moves a pointer when the cursor keys are used, and selects the item next to the pointer when Return is pressed. To make this work with a joystick JStk1 should be *EXECed on to the program and the lines above changed to:

```
100 REPEAT
110 key%=INKEY(4)
120 IF FNl(key%) THEN PROCmoveX(-1)
130 IF FNr(key%) THEN PROCmoveY(1)
140 IF FNd(key%) THEN PROCmove(1)
150 IF FNu(key%) THEN PROCmove(-1)
160 UNTIL FNs(key%)
```

The first point to note is that GET has been changed to INKEY(4). Leaving it as it was would have meant that the joystick would only have been read after a key had been pressed - not particularly useful. The value 4 has been chosen for the time limit rather than zero because otherwise the joysticks may become too sensitive and the pointer start zooming all over the place.

Running Programs with a Joystick

Secondly, tests on *key%* have been replaced by calls to functions with *key%* as the parameter. Information on which tests to change to which function calls are shown in Table 1.

| | KEY | ASCII CODE | FUNCTION |
|--------|----------------|------------|----------|
| LEFT | <Left Cursor> | 136 | FNl() |
| RIGHT | <Right Cursor> | 137 | FNr() |
| DOWN | <Down Cursor> | 138 | FNd() |
| UP | <Up Cursor> | 139 | FNu() |
| SELECT | <Return> | 13 | FNs() |
| | <Space> | 32 | |

Table 1. Functional equivalents using GET

When converting games to run with a joystick, JStk2 will be more useful. Games programs often contain lines like:

```
100 REPEAT
110 IF INKEY-65 THEN PROCman(-1)
120 IF INKEY-2 THEN PROCman(1)
130 IF INKEY-74 THEN PROCfire
140 UNTIL FNdead
```

These could be rewritten as follows if used with the joystick.

```
100 REPEAT
110 IF FNl2 THEN PROCman(-1)
120 IF FNr2 THEN PROCman(1)
130 IF FNf THEN PROCfire
140 UNTIL FNdead
```

Negative INKEY calls have simply been replaced by function calls. A problem may arise here as there isn't a standard set of keys used in games. JStk2 copes with this by testing for several different keys as well as the joystick. Table 2 shows which keys can be used, and which negative INKEY calls should be changed to what. Note that any of the keys shown in Table 2 may be used in a converted game, not just the ones used before the conversion.

A further useful function is FNw. Often programs wait for you to "Press Space to continue". FNw will test for Space or Fire.

```
100 REPEAT UNTIL GET=32
110 REPEAT UNTIL GET$=" "
120 A%=GET
130 A$=GET$
140 REPEAT UNTIL INKEY(0)=32
150 REPEAT UNTIL INKEY$(0)=" "
160 REPEAT UNTIL INKEY-99
```

All the above lines (and there are a few more examples not given) do a similar job in that the computer will pause, continuing when Space is pressed. Any of the lines could be replaced with the following:

```
100 REPEAT UNTIL FNw
to allow <Fire> to be used to continue the
program as well. FNw can be used with
programs reading the keyboard directly
or via the keyboard buffer as it is
contained in Jstk1 (note that it has no
parameter) and Jstk2.
```

| | KEY | -INKEY CODE | FUNCTION |
|---------------|------------------------------|------------------|----------|
| LEFT | Z <Caps Lock> | -98 -65 | FNl2 |
| RIGHT | X <Ctrl> | -67 -2 | FNr2 |
| DOWN | / (?) , (>) | -105 -104 | FNd2 |
| UP | : (*) ; (+) | -73 -88 | FNu2 |
| FIRE/ JUMP | <Return> <Shift> <Tab> | -74 -1 -97 | FNf |

Table 2. Functional equivalents of INKEY

IN PRACTICE

You should by now have some idea of how to use the functions given. Both sets of functions need some searching through the program to be converted. Master users can use the LIST IF command to help them, but BEEBUG's Edikit ROM and BBC Soft's Toolbox with its Cross-

Running Programs with a Joystick

Referencer and Replacer programs are also useful here. Alternatively you could *SPOOL the program and load it as a text file in to a word processor with a search and replace option.

As an example, this is how to alter Glynn Clements' BEEBUG Magazine Disc Menu so that the joystick can be used to make selections:

1. LOAD the program "Menu" and *EXEC JStk1 on to it.
2. Search through the program for references to INKEY and GET, and find that line 1980 reads:
1980 Q%=INKEY(0)
3. Change this to:
1980 Q%=INKEY(4)
4. Search the rest of that function for references to Q% and find the following lines:
2000 IF Q%=138 THEN PROCmove(1)
2010 IF Q%=139 THEN PROCmove(-1)
2020 UNTIL Q%=13
5. Change these lines to:
2000 IF Fnd(Q%) THEN PROCmove(1)
2010 IF FNu(Q%) THEN PROCmove(-1)
2020 UNTIL FNs(Q%)
6. Resave the complete program.

CONCLUSION

I hope you followed through the above example, which many of you will find practical if you have any recent BEEBUG magazine discs. So now, even if you have little experience at all of programming, you can go out and convert many previously published Basic programs to work with the joystick.

If you do have any knowledge of programming then you could try coping with that "awkward" game that uses the "wrong" keys, or change the keys used in the functions to your personal preferences. You might also want to change text messages in programs (e.g. on title screens, or after you've loaded in some data or

been killed) to indicate that the joystick may be used. If you can't see how to do this for a particular program, however, it doesn't in any way affect its running.

Happy joystick using!

Note: the issues of BEEBUG referred to in the article are no longer available. Photocopies of the relevant articles can be provided for anyone sending an A5 (or larger) SAE and £1.00 to cover costs.

Listing 1

```
42 REM Joystick Control by Simon Myer
s
32000 DEFFN1 (K%)=(ADVAL1DIV16>3072ORK%=1
36)
32010 DEFFNr (K%)=(ADVAL1DIV16<1024ORK%=1
37)
32020 DEFFNd (K%)=(ADVAL2DIV16<1024ORK%=1
38)
32030 DEFFNu (K%)=(ADVAL2DIV16>3072ORK%=1
39)
32040 DEFFNs (K%):LOCALX%,Y%:X%=ADVAL1DIV
16:Y%=ADVAL2DIV16:=((ADVAL0AND3)MOD2=1AND
DX%<3072ANDX%>1024ANDY%>1024ANDY%<3072OR
K%=13ORK%=32)
32050 DEFFNw:LOCALX%,Y%:X%=ADVAL1DIV16:Y
%=ADVAL2DIV16:=((ADVAL0AND3)MOD2=1ANDX%<
3072ANDX%>1024ANDY%>1024ANDY%<3072ORINKE
Y-99)
```

Listing 2

```
42 REM Joystick Control by Simon Myer
s
32050 DEFFNw:LOCALX%,Y%:X%=ADVAL1DIV16:Y
%=ADVAL2DIV16:=((ADVAL0AND3)MOD2=1ANDX%<
3072ANDX%>1024ANDY%>1024ANDY%<3072ORINKE
Y-99)
32100 DEFFN12=(ADVAL1DIV16>3072ORINKEY-9
8ORINKEY-65)
32110 DEFFNr2=(ADVAL1DIV16<1024ORINKEY-6
7ORINKEY-2)
32120 DEFFNd2=(ADVAL2DIV16<1024ORINKEY-1
05ORINKEY-104)
32130 DEFFNu2=(ADVAL2DIV16>3072ORINKEY-7
3ORINKEY-88)
32140 DEFFNf=((ADVAL0AND3)MOD2=1ORINKEY-
74ORINKEY-1ORINKEY-97)
B
```

Simulation Modelling (2)

by Mike Williams

This month I want to illustrate the event centred approach to simulation modelling. But first, let me recap on the simple example which we introduced last time.

Customers arrive to be served at intervals determined by sampling from a normal distribution. The time to service each customer is found by sampling from a second distribution. The object of the exercise is to build a model which will allow us to investigate this over some predetermined period of time. In particular, we wish to examine how queue length varies depending on the two distributions from which we sample.

The basic model for an event-centred simulation is shown in figure 1. Within the main loop, we determine the time of the next due event, advance the clock to this time, and then perform actions related to the type of event which has occurred. The loop is repeated until the simulation is complete.

In our example, there are two events:

1. Customer arrives
2. A service is completed

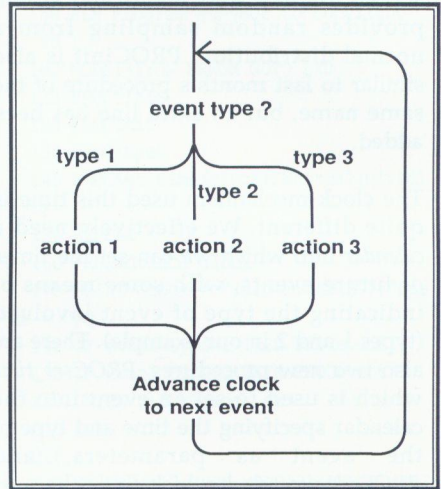


Figure 1. Typical event-based simulation model

The actions which are to be performed for each type of event can be represented as follows:

Event type 1

```

IF queue=0 AND server idle THEN
    Set server busy
    Set end-service time
ELSE
    Increment queue length
    Set next arrival time
  
```

Event type 2

```

IF queue>0 THEN
    Decrement queue length
    Set end-service time
ELSE
    Set server idle
  
```

A program implementing this event-centred approach to our problem is given in listing 1. This copies a number of

Workshop - Simulation Modelling

routines directly from last month's program, those like PROCtitle, PROCinput etc. relating to the general screen display, and FNnormal which provides random sampling from a normal distribution. PROCinit is also similar to last month's procedure of the same name, but an extra line has been added.

The clock mechanism used this time is quite different. We effectively need a *calendar* into which we can set the times of future events, with some means of indicating the type of event involved (types 1 and 2 in our example). There are also two new procedures, *PROCset_time* which is used to set an event into the calendar specifying the time and type of the event as parameters, and *PROCadvance_clock* which finds the next due event in the calendar and advances the clock to this time. The two event types each have a single procedure to themselves.

The calendar is implemented as two arrays, one called *Cal* into which the times of future events are entered, and one called *Event* which records the corresponding event type. An empty entry is denoted by a zero entry in *Cal*(), which is the default state. The arrays need to be large enough to store all the events which might arise during the course of a simulation, and a suitable value is assigned to the variable *TN%* in line 1060. This makes it easy to adjust this value if and when necessary.

When a time is to be set in the calendar, the respective procedure looks for the first non-zero entry in the array *Cal*(), and uses this for the time, placing the event type in the corresponding element of the array *Event*().

The procedure *PROCadvance_clock*, searches through the array *Cal*() for the smallest non-zero entry, and sets the clock to this time and the variable *event_type* to the corresponding type for that event. The entry in *Cal*() is then reset to zero to delete that entry from the calendar. The two event-related procedures, *PROCevent1* and *PROCevent2*, simply follow the pseudo code given before. If you run the program with similar input values to those you used with last month's program you should get the same results (within the variations of random sampling).

I prefer the event-centred approach described above for simulation modelling, and this will be used for any further examples we consider. Last month I left you with the question of how to modify the model to allow the clock to continue running until the last customer had been served. This can be easily achieved in the event-based model by changing lines 200 and 1460, and for completeness adding a further line 205 as listed below:

```
200 UNTIL cclock>clock_end AND q=0
205 PRINT"Clock = ";clock;TAB(20);"que
ue = ";q
1460 IF cclock<=clock_end PROCset_time(c
lock+FNnormal(m1,sd1),1)
```

This ensures that the model continues to run until both the stop time has been reached and queue length is zero. In addition, once the specified duration is complete no further events of type 1 are generated (the reason for modifying line 1460).

Another variation which you might like to consider is to increase the number of servers in the model. The simplest approach is to assume that customers

enter a single queue, leaving the queue when a server becomes free. To implement this in our program we need to replace the variable *server_idle*, with values either TRUE or FALSE, with a new variable *servers* which is initially set to the number of available servers. Each time a server is set to work, the number of servers is reduced by 1, and when an end_service event occurs, the number of servers is increased by 1.

To do this change line 1050 to read:

```
1050 clock=0;q=0:servers=3
```

(you can set whatever number of servers you wish). The two event-based procedures should be recoded as:

```
1440 DEF PROCevent1
1450 IF q=0 and servers>0 THEN servers=
servers-1:PROCset_time(clock+FNnormal(m2,
sd2),2) ELSE q=q+1
1460 IF clock<=clock_end THEN PROCset_ti
me(clock+FNnormal(m1,sd1)1)
1470 ENDPROC
1480:
1490 DEF PROCevent2
1500 IF q>0 THEN q=q-1:PROCset_time(cloc
k+FNnormal(m2,sd2), 2) ELSE servers=serve
rs+1
1510 ENDPROC
```

If you try this out you might like to improve the statistics gathered by your model. How many customers does each server deal with? How long does each server spend idling? You might also like to raise the scale of the model by allocating customers to individual queues, one per server, choosing, say, the shortest queue each time. Then show how the lengths of the queues vary over time.

That's all we have room for this month. We will have a look at some more complex examples next time.

```
10 REM Program Event1
20 REM Version B1.0
30 REM Author Mike Williams
40 REM BEEBUG November 1991
50 REM Program subject to copyright
60 :
100 MODE131:ON ERROR GOTO 240
110 PROCtitle
120 PROCinit
130 PROCinput
140 PROCset_time(clock+FNnormal(m1,sd1
),1)
150 REPEAT
160 PROCadvance_clock
170 PRINT"Clock = ";clock;TAB(20);"que
ue = ";q
180 IF event_type=1 THEN PROCevent1
190 IF event_type=2 THEN PROCevent2
200 UNTIL clock>clock_end
210 END
220 :
230 IF ERR<>17 THEN MODE131:REPORT:PRI
NT" at line ";ERL
240 END
250 :
1000 DEF PROCtitle
1010 PRINTTAB(10,1)"S I M U L A T I O N
M O D E L"
1020 ENDPROC
1030 :
1040 DEF PROCinit
1050 clock=0;q=0:server_idle=TRUE
1060 arrival=0:end_service=0
1065 TN%=10:DIM Cal(TN%),Type(TN%)
1070 ENDPROC
1080 :
1090 DEF PROCinput
1100 INPUTTAB(5,4)"Duration: " clock_en
d
1110 PRINT"Arrival times"
1120 INPUTTAB(5)"Mean: " m1
1130 INPUTTAB(5)"Standard deviation: "
sd1
1140 PRINT"Service times"
1150 INPUTTAB(5)"Mean: " m2
1160 INPUTTAB(5)"Standard deviation: "
sd2
```

Workshop - Simulation Modelling

```
1170 ENDPROC
1180 :
1190 DEF FNnormal (M, SD)
1200 LOCAL I%, X: X=0
1210 FOR I%=1 TO 12
1220 X=X+RND(1)
1230 NEXT
1240 =INT (M+SGN (RND (1)-0.5) *(X-6) *SD+0.5)
1250 :
1260 DEF PROCset_time (time, type)
1270 LOCAL I%: I%=-1
1280 REPEAT: I%=I%+1: UNTIL Cal (I%)=0
1290 Cal (I%)=time: Type (I%)=type
1300 ENDPROC
1310 :
1320 DEF PROCadvance_clock
1330 LOCAL I%, Ltime, LI%: I%=-1
1340 REPEAT: I%=I%+1: UNTIL Cal (I%)>0
1350 Ltime=Cal (I%): LI%=I%: I%=I%+1
1360 REPEAT
```

```
1370 IF Cal (I%)>0 AND Cal (I%)<Ltime THEN
N Ltime=Cal (I%): LI%=I%
1380 I%=I%+1
1390 UNTIL I%>TN%
1400 clock=Ltime: event_type=Type (LI%)
1410 Cal (LI%)=0: Type (LI%)=0
1420 ENDPROC
1430 :
1440 DEF PROCevent1
1450 IF q=0 AND server_idle THEN server
_idle=FALSE: PROCset_time (clock+FNnormal (
m2, sd2), 2) ELSE q=q+1
1460 PROCset_time (clock+FNnormal (m1, sd1
), 1)
1470 ENDPROC
1480 :
1490 DEF PROCevent2
1500 IF q>0 THEN q=q-1: PROCset_time (clo
ck+FNnormal (m2, sd2), 2) ELSE server_idle=
TRUE
1510 ENDPROC
```

B

Acorn Launch A5000 System (continued from page 9)

The A3000 will continue to provide excellent entry level capability for those who prefer its more compact nature and do not need a more expandable configuration, while Acorn says that the flagship A540 will continue to be targetted at the high performance market to which its expansion capability to 16 Mbytes and standard SCSI interface are well suited.

At the same time as launching the A5000, Acorn is also providing an A5000 Learning Curve package, which bundles in 1st Word Plus (word processor), Genesis Plus (multi-format information system), multi-tasking PC Emulator with DR DOS, and Acorn DTP at less than £40 more than the basic A5000 price. This represents excellent value, though many existing users will no doubt wince at the continued promotion of Acorn DTP in

particular when there are such superior third party products available.

Acorn says that in due course (sometime next year) RISC OS 3 will also be made available for existing Archimedes machines: the current version of RISC OS 3 only supports A5000 hardware, and will not work with A300 or A400 machines. At present we do not know the cost of any such upgrade, nor the extent to which the higher demands of RISC OS 3 will work satisfactorily with lower performance ARM2 machines.

For now, let us rest with the fact that Acorn has produced a remarkable system for the price. Let us hope for all our sakes that the rest of the computer world agrees.

B

A Printer Spooler Utility

by D.S. Peckett and C.J.Dawson

Printing is a time consuming activity that all too often can tie up your micro for a long period of time. We present a printer spooler, originally from Vol.3 Nos.3 and 7, which will allow you to carry on working and do your printing at the same time, and it has now been improved and updated to work on the Master series as well.

INTRODUCTION

It is a fact of computing life that a system is never quite fast enough. The Beeb runs at a quite acceptable speed, but as soon as you start printing, your whole system is tied up while it generates a listing of your latest magnum opus. Wouldn't it be nice to be able to carry on using the computer while it was printing?

"Professional" computers (i.e. those costing several thousand pounds plus) often incorporate a "spooler", which is a system allowing the computer to print a file while it carries on with its normal processing. An alternative is to purchase a piece of hardware called a buffer for your printer, which allows large amounts of text to be transferred quickly to the printer, which then deals with this at its own speed, leaving the computer free for other work. But this all costs money, and the small buffer built into many printers will hold very little text (even a 2K buffer is not that large).

This article describes a program which will add a software spooler to a BBC micro, allowing it to print text as a background task while simultaneously processing a totally different program as a foreground task. The program will work with all printers connected via the

printer port (this is the normal practice) but will not work with printers that use the RS423 connector, to avoid potential conflicts with other programs.

The program occupies only two pages of memory and, once loaded, can be used repeatedly. It makes use of the computer's "event" system in order to interleave time allocated to spooling and time allocated to the foreground task in a way that makes both appear to take place concurrently. It is capable of running at the same time as other event-driven routines, though this is partly dependent on how they are written.

SETTING UP THE SPOOLER

As with all programs which involve machine code, take great care when typing the program in, and make sure that you save it before running it in case the program should become corrupted. If you want to save some time when typing in the program, you can omit any text that follows a "\" character, as these are only comments.

Line 150 specifies where in memory the program is to go by calling the procedure FNcode. If you find that the area allocated in the program is not suitable (possibly because a ROM in your machine uses the allocated page for its workspace) then the program can be assembled to any address of your choice by altering line 150 to a specific memory address (e.g. `code%=&A00`).

When you set up the program, you must also set the value of the variable *linefeed* at line 100 to fit your printer's requirements. Set it to FALSE if your

A Printer Spooler Utility

printer provides its own linefeeds after a carriage return, or to TRUE if the computer has to supply it. If you normally need a *FX6,0 before you can print, your printer needs the extra linefeed.

Once you have entered and debugged the routine, you can save it directly as a machine code file. To do this, run the program and then *SAVE it using the parameters as specified by the program itself (you can use an alternative to the name given if you wish). When you load it in future, you only need to type either:

*LOAD *name*

if you want to be able to CALL the spooler from within a Basic program, or:

*RUN *name* OR **name*

if you just want to dump a single text file, where *name* is the name you used when you *SAVED the assembled program.

USING THE SPOOLER

The spooler always assumes that it is going to print a formatted (or pure ASCII) file; it therefore cannot cope with View, Wordwise or Basic listings if there are any extra codes included such as formatting codes or Basic tokens. The way to overcome this is to spool out the text to a file, and then to print that file. With Basic this is simply achieved by using the *SPOOL command before listing the program. This creates a file of pure ASCII codes which can be printed with no problems. Wordwise and InterWord provide a spool option, and this does the job. With View, you need to type *SPOOL and a file name, then SCREEN the appropriate file and *SPOOL again. Having done this you will have a file containing the text correctly formatted, but there will also be the View command mode information. If you wish to delete this, then still using

View you need to type in NEW, READ filename, delete the unwanted headings, and then SAVE filename. The file can then be printed using the spooler.

Once you have set up and saved the text you wish to print, load the spooler program (if it's not in memory already) and start it running by using a CALL command, calling to the address &20 bytes beyond the start of the code. The program will automatically assemble the code to an address determined by the machine on which it is running. This address will be displayed as part of the *SAVE parameters when assembly is complete. For example, if the address shown is &B00, then use:

```
CALL &B20
```

If you have assembled the program into memory by running the original program (as listed), rather than loading it from disc as a machine-code file, you have another way of calling the spooler. During assembly the computer is set up so that you can call the routine directly by the command:

```
*CODE
```

The advantage is that you can run the program from within other languages and applications. These usually allow you to give "*" commands but do not necessarily allow CALLS.

Whichever way you choose to call the spooler, the program will ask you for the name of the file you wish to print - enter it in the usual way and leave the spooler to do its job. You can carry on using your computer in almost any way you wish while the printer mutters away in the background. Every so often, the spooler will read another block of data from the file; while it is doing this, the foreground program will freeze, starting up again as soon as the next block of text has been

loaded. The intermittent freezing is hardly noticeable.

A point to note is that when the routine is running it ignores any presses of the Escape key; this allows you to run programs which use this key for their own purpose. To stop the spooler in mid-print, you must press the Break key.

PROGRAM DESCRIPTION

The routine is driven by the *vertical sync* event. The spooler should be compatible with the vast majority of programs, both Basic and machine code. If you want to know more about events, I thoroughly recommend the article in BEEBUG Vol.7 No.6. The spooler, written in assembler, is contained in the procedure **PROCassem**. This sets up all the constants and variables used by the spooler before determining the name of the file to be spooled.

Using the event mechanism, the spooler is called 50 times a second, when it checks to see whether the next character can yet be sent to the printer. There may also be an occasional pause as the filing system loads the next data block from tape or disc, but this is unavoidable. This delay occurs when all of the data from the current block has been written to the printer. Once spooling is complete, the spool file is closed and the event mechanism disabled. The spooler will then remain dormant until called again. The program is well annotated throughout for those who are interested in its workings.

An interesting point is that whenever Wordwise prepares a file with option 8 (spool), it inserts the value &02 for all of the control codes present within the text file. The spooler checks for these, and sends out a null (&00) byte whenever it

encounters an &02. If Wordwise generates one of its own errors (e.g. markers or room error) then the spooler will stop operating. This is unavoidable, and is due to the way in which Wordwise works.

There is also a check for the "E" character, and the user is able to select the character sent to the printer if this is found. Line 130 should be altered to cater for your own printer; as it stands, it is tailored to an Epson printer.

```
10 REM Program Spooler
20 REM Version B1.1
30 REM Author David Peckett
40 REM Updates C.J.Dawson
50 REM Parallel printers only
60 REM BEEBUG November 1991
70 REM Program subject to Copyright
80 :
90 REM set linefeed to TRUE if printe
r needs linefeeds
100 linefeed=FALSE
110 :
120 REM set ASCII code for printer "po
und"
130 POUND=129
140 :
150 master%=FNmachine:code%=FNcode
160 PROCassem
170 :
180 REM Setup *CODE entry point
190 ?userV=init MOD 256
200 userV?1=init DIV 256
210 :
220 PRINT"Printer spooler installed."
230 PRINT"*CODE will start the spooler
when the"
240 PRINT"Basic program has just been
RUN."
250 PRINT"Use *SAVE SPOOLER ";~code%;"
";~P%+7;" ";~init;" to save"
260 PRINT"the spooler, and then *RUN P
RINT"
270 PRINT"will load and start the spoo
ler."
```

A Printer Spooler Utility

```
280 END
290 :
300 DEF PROCassem
310 :
320 REM Operating system addresses
330 osascii=&FFE3:osbget=&FFD7
340 osbyte=&FFF4:osfind=&FFCE
350 osword=&FFF1:evntv=&220
360 userv=&200:findv=&21C
370 bgetv=&216:bputv=&218
380 :
390 REM Printer VIA addresses
400 ora=&FE61:pcr=&FE6C
410 ifr=&FE6D:ier=&FE6E
420 :
430 name=code%:REM Space for file name
440 namelen=12:REM maximum length of filename
450 bufferparams=code%+namelen+1:REM Buffer for name input
460 :
470 REM Set up osword 0 parameter block
480 bufferparams!0=name
490 bufferparams?2=namelen
500 bufferparams?3=32
510 bufferparams?4=126
520 :
530 REM Program variables
540 channel=bufferparams+5
550 tempier=channel+1
560 tempocr=tempier+1
570 old_evntv=tempocr+1
580 needlf=old_evntv+2
590 acccon=needlf+1
600 old_findv=acccon+1
610 old_bgetv=old_findv+2
620 old_bputv=old_bgetv+2
630 spoolflag=old_bputv+2
640 :
650 FOR pass=0 TO 2 STEP 2
660 P%=spoolflag+1
670 [OPT pass
680 :
690 .init
700 \ store current vectors
710 LDA bgetv:STA new_bgetv+1
720 LDA bgetv+1:STA new_bgetv+2
```

```
730 LDA bputv:STA new_bputv+1
740 LDA bputv+1:STA new_bputv+2
750 :
760 LDY #0
770 STY spoolflag:STY needlf
780 :
790 .type \ Print message
800 LDA message,Y
810 BEQ getfilename
820 JSR osascii
830 INY
840 BNE type
850 :
860 .getfilename
870 LDA #0
880 LDX #bufferparams MOD 256
890 LDY #bufferparams DIV 256
900 JSR osword
910 BCC openfile \ Test for ESCAPE
920 RTS \ Exit if ESCAPE
930 :
940 .openfile
950 LDA #&40
960 LDX #name MOD 256
970 LDY #name DIV 256 \ Point to name
980 JSR osfind \ Open file
990 STA channel
1000 BNE spoolstart \ Did it open?
1010 LDA #7:JSR osascii:RTS \ Return if not
1020 :
1030 \ Add code to event vector
1040 .spoolstart
1050 PHP:SEI \ Mask interrupts
1060 LDA ier
1070 STA tempier \ Save VIA mode..
1080 LDA pcr
1090 STA tempocr \..and handshake
1100 LDA #&7F
1110 STA ier \ Disable VIA interrupts
1120 LDA #&0A
1130 STA pcr \ Set handshake mode
1140 LDA ora \ Dummy to start system
1150 :
1160 \ Save current event vector
1170 LDA evntv:STA old_evntv
1180 LDA evntv+1:STA old_evntv+1
1190 :
```

```

1200 \ intercept event vector
1210 LDA #event MOD 256:STA evtv
1220 LDA #event DIV 256:STA evtv+1
1230 :
1240 \intercept other vectors
1250 LDA findv:STA old_findv
1260 LDA findv+1:STA old_findv+1
1270 LDA #new_findv MOD 256:STA findv
1280 LDA #new_findv DIV 256:STA findv+1
1290 LDA bgetv:STA old_bgetv
1300 LDA bgetv+1:STA old_bgetv+1
1310 LDA #new_bgetv MOD 256:STA bgetv
1320 LDA #new_bgetv DIV 256:STA bgetv+1
1330 LDA bputv:STA old_bputv
1340 LDA bputv+1:STA old_bputv+1
1350 LDA #new_bputv MOD 256:STA bputv
1360 LDA #new_bputv DIV 256:STA bputv+1
1370 :
1380 \ Enable vertical sync Event
1390 LDA #14:LDX #4:JSR osbyte
1400 PLP \ Restore previous interrupt s
tatus
1410 RTS
1420 :
1430 \ Actual spooler routine
1440 :
1450 .event
1460 PHP:CMP #4:BEQ over:JMP not4:.over
PHA:]
1470 IF NOT master% THEN 1490
1480 [OPT pass:LDA &FE34:STA acccon:AND
#&F7:STA &FE34:]
1490 [OPT pass:\ Monitor Printer VIA
1500 LDA ifr \ Check printer flag
1510 AND #2 \ Handshake complete?
1520 BNE over1:JMP busy \ If not, no ac
tion
1530 .over1 TXA:PHA:TYA:PHA
1540 LDA #0:CMF spoolflag
1550 BEQ disable:JMP printit
1560 :
1570 .disable
1580 \ Disable Event 4 whilst accessing
file
1590 \ avoid OSBYTE calls in event code
1600 LDA #0:STA &2C3
1610 ]
1620 :

```

```

1630 REM Include appropriate code for p
rinter
1640 IF linefeed THEN PROCaddlf ELSE PR
OCnolf
1650 :
1660 REM Resume normal assembly
1670 :
1680 [OPT pass
1690 :
1700 .nolinefeed
1710 \ Re-enable Event 4
1720 \ avoid OSBYTE calls in event code
1730 LDA #14:STA &2C3:BNE exit
1740 :
1750 .finished
1760 \ Event 4 is always disabled
1770 \ at this point
1780 PHP:SEI \mask interrupts
1790 LDA old_findv:STA findv
1800 LDA old_findv+1:STA findv+1
1810 LDA #0:LDY channel
1820 JSR osfind \Close file
1830 LDA tempier
1840 STA ier \ Restore
1850 LDA tempocr \ VIA
1860 STA pcr \ registers
1870 \ Restore vectors
1880 LDA old_evtv:STA evtv
1890 LDA old_evtv+1:STA evtv+1
1900 LDA old_bgetv:STA bgetv
1910 LDA old_bgetv+1:STA bgetv+1
1920 LDA old_bputv:STA bputv
1930 LDA old_bputv+1:STA bputv+1
1940 PLP \ Restore previous interrupt s
tate
1950 :
1960 .exit
1970 PLA:TAY:PLA:TAX
1980 .busy:]
1990 IF NOT master% THEN 2010
2000 [OPT pass:LDA acccon:STA &FE34:]
2010 [OPT pass:PLA
2020 .not4 PLP \ Restore system
2030 JMP (old_evtv) \ Continue Event h
andling
2040 :
2050 .new_findv \test for close files
2060 CMP #0:BNE new_findv_out \branch i

```

A Printer Spooler Utility

```
f not close
2070 CPY #0:BNE new_findv_out \close all
l files?
2080 RTS
2090 .new_findv_out \not close all file
s
2100 JMP (old_findv) \call old vector
2110 :
2120 .new_bgetv PHA:LDA #1:STA spoolfla
g:PLA
2130 .new_bgetv1 JSR &FFFF:PHA:LDA #0
2140 STA spoolflag:PLA:RTS
2150 :
2160 .new_bputv PHA:LDA #1:STA spoolfla
g:PLA
2170 .new_bputv1 JSR &FFFF:PHA:LDA #0
2180 STA spoolflag:PLA:RTS
2190 :
2200 \ Message for file opening
2210 .message EQU$7+"File? "+CHR$0
2220 ]
2230 NEXT
2240 ENDPROC
2250 :
2260 REM Code for when extra linefeed n
eeded
2270 DEF PROCaddlf
2280 [OPT pass
2290 LDA needlf \ linefeed required thi
s time?
2300 BEQ getch \ If not, continue
2310 LDX #0:STX needlf \ Clear flag
2320 BEQ nogetch
2330 :
2340 .getch
2350 \ Read a char from file
2360 LDY channel
2370 JSR osbget \ Get next char
2380 BCS finished \ EOF?
2390 :
2400 \patch Wordwise codes
2410 CMP #2:BNE test:LDA #&20:JMP print
it
2420 \patch pound symbol
2430 .test CMP #ASC""":BNE nogetch
2440 LDA #POUND
2450 :
2460 .nogetch
```

```
2470 .printit STA ora \ Output to print
er
2480 CMP #13 \ Was it a CR?
2490 BNE nolinefeed \ If not, continue
2500 LDA #10:STA needlf \ Set flag
2510 ]
2520 ENDPROC
2530 :
2540 REM Code for printers with autolin
efeed
2550 DEF PROCnolf
2560 [OPT pass
2570 \ Get next char
2580 LDY channel
2590 JSR osbget
2600 BCS finished \ EOF?
2610 :
2620 \patch Wordwise codes
2630 CMP #2:BNE test:LDA #&20:JMP print
it
2640 \patch pound symbol
2650 .test CMP #ASC""":BNE printit
2660 LDA #POUND
2670 :
2680 \ Output to printer
2690 .printit STA ora
2700 ]
2710 ENDPROC
2720 :
2730 DEF FNcode
2740 LOCAL A%,Y%
2750 A%=0:Y%=0
2760 A%=USR&FFDA AND &FF
2770 IF A%=1 OR A%=2 Y%=&C00
2780 IF A%=4 OR A%=8 Y%=&900
2790 IF master% Y%=&B00
2800 IF Y%=0 PRINT!""The spooler is desi
gned to work with!""tape or disc filing
systems only!"":END
2810 =Y%
2820 :
2830 DEFFNmachine
2840 REM returns TRUE if machine is Mas
ter series
2850 LOCAL A%,X%,Y%,M%
2860 A%=&81:X%=0:Y%=255
2870 M%=(USR&FFF4 AND &FF0) DIV &100
2880 =(M%=&FD OR M%=&F5)
```





Program Logic

Alan Wrigley looks at the way in which Basic evaluates expressions.

When we speak a sentence such as "If there are the same number of apples as bananas, then I'm a Dutchman", we rarely think about the *precise* meaning of what we've said. We are comparing two quantities, and the important factor in our mind is whether we perceive equality. In a computer language such as Basic, the same comparison, and the consequence of the result, could be expressed as:

```
IF A=B THEN I=D
```

However, there is a very crucial difference between the way the human mind sees the expression, and the way it is seen by a computer. Realising this is the key to understanding program logic, and is essential if you are going to create sound, well-structured programs. Tracking down program errors caused by faulty logic can sometimes be very difficult, particularly if you don't fully understand the logic you are using in the first place.

As I have said, the important factor for us is the concept of equality, or of inequality if that is the case. In other words, the brain evaluates an expression such as:

```
IF A=B
```

by considering whether A is equal to B, and similarly:

```
IF A>B
```

by considering whether A is greater than B. So we take into account the number of apples in the bowl and the number of bananas in the bunch, make a value judgement about the equality or inequality, and base our actions on whether we see equality or not.

The Basic language in your computer, however, looks at the expression in a

completely different way. For Basic, the only concepts that are relevant are truth and falsehood. If an expression is true, one course of action is taken, if it is false, another. So the expression:

```
IF A=B
```

is actually evaluated by Basic as:

```
IF (A=B)=TRUE
```

At this point we need to understand exactly what is meant by TRUE and FALSE. These concepts are represented in the computer by the values -1 and 0 respectively, so if you type:

```
PRINT TRUE
```

you will get the answer -1, and similarly PRINT FALSE will produce 0. When you assign TRUE or FALSE to a variable, these are the values that will be used. Similarly when an expression is evaluated which does not produce a numerical result (for example A=B above), the same values of -1 and 0 are used. So the following two lines of code:

```
A=5:B=5
```

```
PRINT (A=B)
```

will give the answer -1.

However, it is perfectly possible to base conditional statements, such as IF or UNTIL, on *any* expression, including those which produce a numerical result. Suppose the expression is 5*5, resulting in a value of 25. This is neither 0 nor -1, and yet I have implied that any expression will be evaluated as true or false. What happens is that, for the purposes of logical evaluation, Basic treats as true any value which is not false, i.e. non-zero, even though it will only ever *assign* -1 as the value of TRUE.

This means, of course, that you can equally well base a conditional statement on the state of a single variable as on a

First Course

comparison between two. Beginners to programming often find this hard to understand; for example, the following line:

```
IF A=B=C
```

appears at first sight to be meaningless.

To understand this, you must first realise that, except in a few special circumstances which we need not detail here, the Basic keyword THEN is optional. Secondly, if we bear in mind what I have just said, the variable A, if it follows an IF statement, will be evaluated as true (if it is not zero) or false (if it is zero). Thus we could write the above line as:

```
IF A=TRUE THEN B=C
```

which now makes much more sense. Equally we could write:

```
IF A<>0 THEN B=C
```

which will produce exactly the same result.

Thus I hope you can see the subtle but crucial difference between human and computer logic - in the former case it is the expression itself which is considered; in the latter case, it is merely whether the expression is true or false.

The ability to act upon the evaluation of a variable is widely exploited by programmers. A typical example would be an input routine where the user may enter any value except zero, as in the following:

```
REPEAT
INPUT "Enter a value" A
UNTIL A
```

where the REPEAT loop which prompts the user for his input will continue to be executed until A, which holds the entered value, is not zero. Another example might be a flag which needs to be set at some point, which will determine whether a particular action is taken somewhere else in the program. The following example demonstrates this:

```
PROCcheck
IF flag THEN PROCsave
```

```
END
....
....
DEF PROCcheck
flag=FALSE
PRINT "Do you want to save your file?"
IF GET=89 THEN flag=TRUE
ENDPROC
```

Here, a call is made to PROCcheck. First of all, the variable *flag* is set to zero. The user is then asked if he wants to save his file. The GET statement evaluates the next key pressed, and if this is Y (ASCII 89) then *flag* is set to TRUE. On returning to the main program, PROCsave is called if *flag* is true.

An even neater way to achieve the same result in this particular case is to use a function to return a value of TRUE or FALSE and then act directly on this, as in the following example:

```
IF FNcheck THEN PROCsave
END
....
....
DEF FNcheck
PRINT "Do you want to save your file?"
=(GET=89)
```

You will find such use of variables and functions throughout many programs. If a program is to be well-structured (i.e. not riddled with GOTOs or other nasties such as jumping out of loops), then there are many situations where the desired result is almost impossible to achieve without such techniques.

COMPLEX EXPRESSIONS

So far we have only considered very simple expressions, such as A=B or even just single variables. However, there are many instances where a more complex expression needs to be evaluated, or where several simple expressions can be merged into one. For example, in the following statement, the expression following IF, for all its complexity, will

still be evaluated as either true or false, nothing more, nothing less:
`IF ((A+B)*(C AND D) OR ((X DIV Y) AND Z))-3`

It is even possible in some cases to do away with conditional statements altogether and perform a calculation by directly incorporating the evaluation of the expression into the calculation itself. For example, consider the statement:

`IF A=B THEN C=C-1`

As I have said, Basic treats this as:

`IF (A=B)=TRUE THEN C=C-1`

Now if we remember that $(A=B)$, being an expression which does not produce a numerical result, is evaluated as either -1 (TRUE) or 0 (FALSE), we can write this much more simply as:

`C=C+(A=B)`

In other words, if $(A=B)$ is true, C is made equal to C plus -1 (i.e. $C=C-1$). If $(A=B)$ is not true, then C is unchanged (i.e. $C=C+0$).

NEGATIVE LOGIC

There are a number of pitfalls for the unwary programmer when dealing with logic. One of these is the use of the keyword NOT. Now this keyword, as you might expect, negates the truth of an expression; in other words, NOT TRUE evaluates as FALSE and vice versa. So if for example a variable, A , might alternate between the two states of TRUE and FALSE, we could use:

`IF A THEN...`

to perform an action if it is true, or:

`IF NOT A THEN...`

if it is false.

But now consider an expression which produces a numerical result, which may not necessarily be either 0 or -1. If we execute the following lines of Basic:

`A=5:B=5`

`IF A*B THEN PRINT "Hello"`

then "Hello" will be printed on the next line, indicating that $A*B$ is true. This is exactly what we would expect, since Basic will treat the result (i.e. 25) as true, since it is non-zero.

Now try the following:


`A=5:B=5`

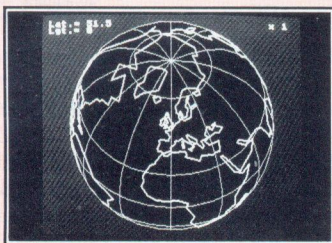
`IF NOT (A*B) THEN PRINT "Hello"`

You will see that "Hello" is still printed. So what has gone wrong? Surely if $A*B$ is true, then NOT $(A*B)$ must be false? The answer lies in the nature of the NOT statement, and also reflects the way in which the computer sees logic, as I described at the beginning of this article.

What NOT does is to toggle all the bits in the value produced by the expression. This is known as negating the number. So if we take the number zero, in which all bits are turned off, and negate it, we get -1, because in the signed arithmetic used by Basic, -1 is represented by all bits set. If we use an 8-bit signed binary number as an example, this would then be 1111111. Thus NOT 0 = -1. Similarly, NOT 1 (00000001) = -2 (11111110) and NOT 25 (00011001), as in our example above, becomes -26 (11100110), which is still treated as true by Basic and so the PRINT statement is executed. What Basic is actually doing is equivalent to:

`IF NOT (A*B)=TRUE THEN...`

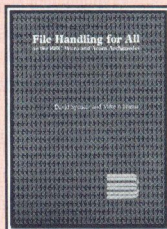
In fact the *only* value which negates to FALSE is TRUE itself, or -1. So you should be very careful not to use NOT in conditional statements unless you are dealing with the actual values of TRUE and FALSE only. It is all too easy to fall into this trap. For instance, I have seen many programs where a variable toggles between 0 and 1, and at some point in the program the desired result fails to materialise because the programmer has used NOT and is expecting the expression to evaluate to FALSE if the variable has a value of 1. You may also sometimes find that you have changed the nature of a variable such that, while it originally toggled between TRUE and FALSE, it may now produce a greater range of values, and the logic fails because you have used NOT. 



Applications I Disc

- BUSINESS GRAPHICS** - for producing graphs, charts and diagrams
- VIDEO CATALOGUER** - catalogue and print labels for your video cassettes
- PHONE BOOK** - an on-screen telephone book which can be easily edited and updated
- PERSONALISED LETTER-HEADINGS** - design a stylish logo for your letter heads
- APPOINTMENTS DIARY** - a computerised appointments diary
- MAPPING THE BRITISH ISLES** - draw a map of the British Isles at any size
- SELECTIVE BREEDING** - a superb graphical display of selective breeding of insects
- PERSONALISED ADDRESS BOOK** - on-screen address and phone book
- THE EARTH FROM SPACE** - draw a picture of the Earth as seen from any point in space
- PAGE DESIGNER** - a page-making package for Epson compatible printers
- WORLD BY NIGHT AND DAY** - a display of the world showing night and day for any time and date of the year

File Handling for All on the BBC Micro and Acorn Archimedes by David Spencer and Mike Williams



Computers are often used for file handling applications yet this is a subject which computer users find difficult when it comes to developing their own programs. *File Handling for All* aims to change that by providing an extensive and comprehensive introduction to the writing of file handling programs with particular reference to Basic.

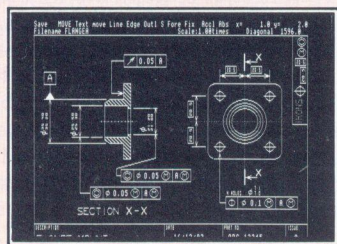
File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File Handling and Databases on the Beeb and the Arc. However, all the file handling concepts discussed are relevant to most computer systems, making this a suitable introduction to file handling for all.

The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical and relational database design, and the book concludes with a chapter of practical advice on how best to develop file handling programs.

The topics covered by the book include:

- Card Index Files, Serial Files, File Headers, Disc and Record Buffering, Using Pointers, Indexing Files, Searching Techniques, Hashing Functions, Sorting Methods, Testing and Debugging, Networking Conflicts, File System Calls

The associated disc contains complete working programs based on the routines described in the book and a copy of *File*, a full-feature Database program originally published in BEEBUG magazine.



ASTAAD

Enhanced ASTAAD CAD program for the Master, offering the following features:

- * full mouse and joystick control
- * built-in printer dump
- * speed improvement
- * STEAMS image manipulator
- * Keystrips for ASTAAD and STEAMS
- * Comprehensive user guide
- * Sample picture files

| | Stock Code | Price | | Stock Code | Price |
|-------------------------------------|------------|--------|------------------------------------|------------|--------|
| ASTAAD (80 track DFS) | 1407a | £ 5.95 | ASTAAD (3.5" ADFS) | 1408a | £ 5.95 |
| EDIKIT (EPROM) | 1451a | £ 7.75 | | | |
| EDIKIT (40/80T DFS) | 1450a | £ 5.75 | EDIKIT (3.5" ADFS) | 1452a | £ 5.75 |
| Applications II (80 track DFS) | 1411a | £ 4.00 | Applications II (3.5" ADFS) | 1412a | £ 4.00 |
| Applications I Disc (40/80T DFS) | 1404a | £ 4.00 | Applications I Disc (3.5" ADFS) | 1409a | £ 4.00 |
| General Utilities Disc (40/80T DFS) | 1405a | £ 4.00 | General Utilities Disc (3.5" ADFS) | 1413a | £ 4.00 |

Please add p&p

Board Games

SOLITAIRE - an elegant implementation of this ancient and fascinating one-player game, and a complete solution for those who are unable to find it for themselves.

ROLL OF HONOUR - Score as many points as possible by throwing the five dice in this on-screen version of 'Yahtzee'.

PATIENCE - a very addictive version of one of the oldest and most popular games of Patience.

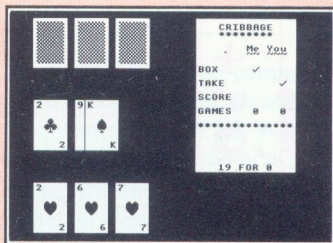
ELEVENSES - another popular version of Patience - lay down cards on the table in three by three grid and start turning them over until they add up to eleven.

CRIBBAGE - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

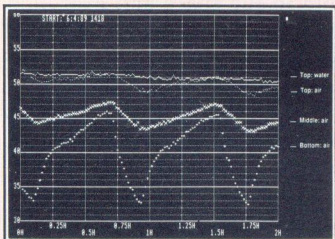
TWIDDLE - a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern.

CHINESE CHEQUERS - a traditional board game for two players, where the object is to move your counters, following a pattern, and occupy the opponent's field.

ACES HIGH - another addictive game of Patience, where the object is to remove the cards from the table and finish with the aces at the head of each column.



Applications III Disc



CROSSWORD EDITOR - for designing, editing and solving crosswords

MONTHLY DESK DIARY - a month-to-view calendar which can also be printed

3D LANDSCAPES - generates three dimensional landscapes

REAL TIME CLOCK - a real time digital alarm clock displayed on the screen

RUNNING FOUR TEMPERATURES - calibrates and plots up to four temperatures

JULIA SETS - fascinating extensions of the Mandelbrot set

FOREIGN LANGUAGE TESTER - foreign character definer and language tester

LABEL PROCESSOR - for designing and printing labels on Epson compatible printers

SHARE INVESTOR - assists decision making when buying and selling shares.

Arcade Games

GEORGE AND THE DRAGON - Rescue 'Hideous Hilda' from the flames of the dragon, but beware the flying arrows and the moving holes on the floor.

EBONY CASTLE - You, the leader of a secret band, have been captured and thrown in the dungeons of the infamous Ebony Castle. Can you escape back to the countryside, fighting off the deadly spiders on the way and collecting the keys necessary to unlock the coloured doors?

KNIGHT QUEST - You are a Knight on a quest to find the lost crown, hidden deep in the ruins of a weird castle inhabited by dangerous monsters and protected by a greedy guardian.

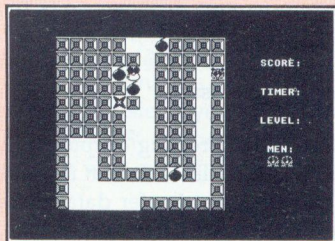
PITFALL PETE - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way.

BUILDER BOB - Bob is trapped on the bottom of a building that's being demolished. Can you help him build his way out?

MINEFIELD - Find your way through this grid and try to defuse the mines before they explode, but beware the monsters which increasingly hinder your progress.

MANIC MECHANIC - Try to collect all the spanners and reach the broken-down generator, before the factory freezes up.

QUAD - You will have hours of entertainment trying to get all these different shapes to fit.



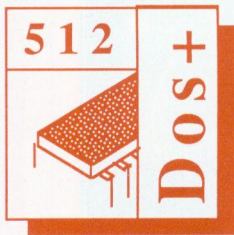
| | Stock Code | Price |
|--|------------|---------|
| Arcade Games (40/80 track DFS) | PAG1a | £ 5.95 |
| Board Games (40/80 track DFS) | PBG1a | £ 5.95 |
| File Handling for All Book | BK02b | £ 9.95 |
| File Handling for All Disc (40/80T DFS) | BK05a | £ 4.75 |
| Joint Offer book and disc (40/80T DFS) | BK04b | £ 11.95 |

| | Stock Code | Price |
|---|------------|---------|
| Arcade Games (3.5" ADFS) | PAG2a | £ 5.95 |
| Board Games (3.5" ADFS) | PBG2a | £ 5.95 |
| File Handling for All Disc (3.5" ADFS) | BK07a | £ 4.75 |
| Joint Offer book and disc (3.5" ADFS) | BK06b | £ 11.95 |

Please add p&p. UK: £1.00 first item (50p for every additional item), Europe and Eire: £1.60 first item (80p every additional item), Elsewhere: £2.60 first item (£1.30 every additional item)

Tel. (0727) 40303

Fax. (0727) 860263



512 Forum

by Robin Burton

Before we leave batch files, the topic which has occupied the

Forum for the past three issues, I have one more tip to pass on.

David Harper, who you'll remember sent a letter about the '/C' switch for COMMAND.COM, also mentioned this item at the same time, but it's had to wait until now for a suitable point to squeeze it in.

HIDING OUTPUT

In Vol.10 No.2 I mentioned that one or two of you had asked how to suppress the screen output from batch files, and I offered some ideas.

David pointed out something else which I should have remembered but didn't, so thanks again to him. My excuse is that these days I don't get much chance to explore or even to use the system unless there's a specific reason.

I've mentioned redirecting output in the Forum a couple of times, but in its usual context, which is to transfer data to real files and devices. In fact, there's also a 'pseudo' device in DOS Plus called 'NUL:' (note that the single 'L' in the spelling is not a mistake.)

You can use 'NUL:' just as you'd use a real file or device name in batch files or in manual commands to re-direct screen or other output to it. This is what I described in Vol.10 No.2. 'NUL:' acts as a printer 'sink', swallowing all output which would otherwise be sent to the printer. For example, using 'NUL:' to

suppress screen output from a copy command the syntax is:

```
COPY FILE1 FILE2 >NUL:
```

The copy works OK, but there's no display. Disadvantages of using 'NUL:' are non-existent. You should remember though, if you're short of disc space and don't want to split your batch file into two jobs as was explained previously, there's more typing. 'NUL:' does offer savings both in file space and execution time, but you'll have to supply the '>NUL:' suffix for every command for which the output is to be suppressed. However, creating batch files is a simple job and it's usually a 'one-off' exercise so this is a small hardship.

The advantages, however, far outweigh this single possible negative. Using a 'real' temporary file, as I explained before, both slows down the job and, if you're being tidy, the temporary file needs deleting afterwards each time too. Using two batch files but employing 'NUL:' in place of a temporary file is a much better idea. The end result is similar to the temporary file, but without the disadvantages.

However, don't get the idea that this facility is the perfect answer for all such problems. As usual there's a caveat.

As I mentioned in Vol.10 No.2, though I didn't explain it fully there, when you re-direct screen output you need to be careful which commands are affected. If the resulting output from a command is something which is intended to change the screen display (basically all commands which are executed in the 512

by means of BBC micro 'VDU' operations), re-directing will cancel the effects of the command. This is still true when you re-direct to 'NUL:' instead of to a file, and it's easy to demonstrate.

Re-direction can be used manually as well as in batch files, so let's take a simple example such as 'CLS' and try it manually. In the 512 (but not in PCs) the similarity of DOS's 'CLS' command to the BBC Basic command is more than just skin deep. Both issue the equivalent of 'VDU12'. First try 'CLS' and the screen clears normally, just to prove I've nothing up my sleeve. If you now type:

```
CLS >NUL:
```

absolutely nothing happens. Well, that's not quite true, but it might as well be. What does happen is that the BDOS passes the output from the 'CLS' command to the XIOS as usual. Next though, instead of the output being directed to the default device 'CON:' within the XIOS (and from there being translated to a VDU12 through the Tube to the BBC micro), it is directed to device 'NUL:', which means it's never physically output anywhere. Try these two example batch files, called TEST.BAT and TEST1.BAT, to see better what happens.

```
ECHO OFF
ECHO THIS IS BATCH FILE 1
REM Call the second file
TEST1 >NUL:
```

```
ECHO THIS IS BATCH FILE 2
REM Now try some other commands
BACKG
CLS
PCSCREEN 7
DIR
COPY TEST.BAT TEMP.FIL
```

You can see what the commands would do 'normally' in the second file, but if you try running the job you'll find that, in effect, all the commands in the second file are pointless with the sole exception of COPY, the only command which is not directly concerned with console output. After the job ends you'll find there is indeed now a TEMP.FIL, which clearly proves that the second file really did execute. This in turn proves that not only the visible screen output, but that all the screen changes have been cancelled by the re-direction to 'NUL:'

The final point is that using 'NUL:' doesn't get round the other problem mentioned last time either. You're still stuck with at least the first command on the screen. In this case the 'ECHO OFF' in TEST.BAT will be displayed (and 'no', you can't have 'ECHO OFF >NUL:', it doesn't work!).

There are some dodges which can minimise this problem: David mentioned one which I might include in a future issue, but in fairness it's a bit fiddly. In practical terms it's almost as effective and much easier if you follow the 'ECHO OFF' by 'CLS' on the next line and hope no-one was watching the screen too closely!

CURIOSITIES

As I mentioned before, I rarely get time just to 'poke around' in the system these days, I'm usually programming or writing. This is a pity, because it can be quite entertaining and it's surprising what you can find.

For example in some jobs, such as writing and checking batch file examples for the Forum, I'm switching between the BBC micro and the 512 repeatedly. As a

result, in these circumstances I occasionally type something like '*.' when what I really mean is 'DIR'. It's not surprising that it doesn't work, but I wonder how many of you have done this sort of thing and found that it doesn't cause an error either. At the DOS prompt type:

```
*This is rubbish
```

and see what happens. Nothing! There's no 'Bad command or filename' and in fact there's no action of any sort. It also works in batch files in exactly the same way, the line behaves as if it's not there, apart from the display if echo is on. If you experiment a bit you'll find that ',', '}', '' and numerous other characters can be used instead of '*' with the same result.

I've known about these oddities for a long time, but I've never found any practical use for them, except that they can be used as REMs in batch files and save the typing of two more characters. Have any of you any ideas?

You may remember that I wrote about piping a while ago, in Vol.9 No.7 to be precise. This is the method by which the output from one process can be fed directly to another as its input, and it uses the vertical bar symbol, '|'.

I discovered a curiosity (a bug?) associated with this a while ago, and you can try this one yourself too. Make sure you have a disc in the current drive, then enter the following command exactly as shown.

```
|This does nothing
```

When you've entered it, the usual DOS prompt returns without an error, but

then try to do something else, or more accurately anything else. You can type into the command line as usual, but no matter what you enter, including any of DOS's built-in commands, the only thing you'll get out of the system from now on is 'File not found'.

The only way out of this is to re-boot, but when you've done that issue a 'DIR' on the disc you were using and take a look at the directory. You should find an extra file, called '%PIPE650.\$\$\$'. If you now 'TYPE' that file you'll find it's empty.

You can certainly get data into the file, and to prove it, next try:

```
|BACKG
```

There'll be no visible output from BACKG and the immediate effect is that you'll end up with the same situation as before: further commands don't work. However, after re-booting this time you'll see, if you 'TYPE %PIPE650.\$\$\$', that the screen output from BACKG was written to the file, giving exactly the same result (apart from the effective hang-up in this case) as:

```
BACKG >%PIPE650.$$$
```

I'll let you play with that one on your own, but I may come back to it in the future.

COMPETITION RESULTS

You'll remember that in 512 Forum, in Vol.10 No.4, we had a competition which closed on September 20th. I didn't think the answers were too hard to work out but I might have mis-judged, because I have to say that I was disappointed by the fact that there were only four correct entries.

The four members concerned each win a £20.00 prize which they can spend on Essential Software products of their choice. The names of the four, including membership numbers are:

R.G. Shrubsall of Newcastle upon Tyne
(membership No. T103435)
S.J. Bowden of London (C111290)
A.D.S. Benham, also of London (C36673)
D.T. Blyth of Epsom, Surrey (C22920)

The answers to the number of files which would be produced in sets 'A' to 'H' in sequence by the example batch file are:

| Set | Files in set | Running Total |
|-----|--------------|---------------|
| A | 8 | 8 |
| B | 36 | 44 |
| C | 120 | 164 |
| D | 330 | 494 |
| E | 792 | 1286 |
| F | 1716 | 3002 |
| G | 3432 | 6434 |
| H | 6435 | 12869 |

As you can see, the number of files in each succeeding set is growing steadily but relentlessly, and that was the reason for the second part of the question, "Would any individual set, up to 'Z', produce a million files?"

The answer is "Yes". Assuming it were a practical possibility, set 'U' would generate 1,184,040 files and the running total would have reached 4,292,144.

As is usual with this sort of problem, the best approach to finding the answer is to understand the way the mechanism that generates the files works, after which it's remarkably simple. The competition didn't ask you for the method you'd used, but R.G. Shrubsall's entry included his approach. It was ingenious and based

purely on numeric theory. He says he thought writing a program wasn't necessary.


I have a programmer's mentality, so naturally I disagree. I thought a short Basic program was the obvious approach. D.T. Blyth thinks the same and included his program too. The other two didn't explain their method, but there weren't any prizes for that so it didn't matter.

I know we're straying into 'BBC' territory somewhat, but if any of you want to see the results for sets 'A' to 'Z' just enter and run this Basic program. It can actually be entered as one line if you like, but for clarity I've split it into five.

```
DIM F%(8) : F%(1) = 1 : O% = 0
FOR S% = ASC "A" TO ASC "Z" : F% = 0
FOR L% = 1 TO 8 : F%(L%) = F%(L%) + F%(L% - 1)
F% = F% + F%(L%) : NEXT
O% = O% + F% : PRINT CHR$(S%), F%, O% : NEXT
```

It will list the answers both for each set, and in total from 'A' to 'Z', in about a second. If you run it in mode 0 (or 128), starting with a clear screen all the answers will be shown without paging.

By way of explanation, a one dimensional array of nine elements, F%() is set up in the first line and is used to calculate the number of files (with names of length L% characters) in each set, S%. F%(1) is initially seeded with 1, representing the original start file, then for each set F% accumulates the total of files produced, while O% accumulates the overall running total.

As you can see, two loops and simple addition is all that was needed. If you initially thought the problem too difficult I hope that now you've seen the answers you don't think it was unreasonable. 

A Timed Interrupt Routine

by Dr. Robin Murphy

Robin Murphy describes a routine which allows two key presses to interrupt another program in order to initiate a machine code program such as a screen dump, or similar.

Printer dump routines for the BBC B and Master machines are now commonplace, (BEEBUG Vol.5 No.5, Vol.7 No.7, Vol.8 No.6), and it is easy to include a call to one in your own programs. For example, at the appropriate stage, add some extra code like:

```
*pdump
```

or something more sophisticated like:

```
IF GET$="P" THEN *pdump
```

which will print the current screen to your printer.

With pre-written software this may not be so easy. Much better would be to make some particular pair of key presses cause the screen to be printed at any stage you choose.

This requires the setting up of some machine code which is executed at regular intervals in order to monitor which keys are currently pressed. This must be done without seeming to affect the computer's current task. This is possible using the Versatile Interface Adaptor (VIA) which also controls the user port.

There are four main stages to the program:

1. Enabling the interrupt and redirecting the Break vector.

Timer 1 of the VIA will be loaded with the number &FFFF, which will produce an interrupt every 65535 microseconds or about 66 milliseconds. The timer will count down to zero, when it will generate an interrupt, execute our interrupt code, and reset the timer once again.

This is done in the assembler subroutine "enable" at lines 1440-1580. It is also necessary to enable the Timer 1 interrupt, and to redirect the IRQ2 vector to point to the interrupt routine.

We must also provide some code to be executed whenever Break is pressed to prevent all this work being cancelled by a Break. This needs the code for a JMP instruction (&4C) to be stored at address &287 followed by the address of a "break" routine.

2. The break code.

This will normally be executed twice whenever Break is pressed; first with the Carry flag CLEAR, then with Carry SET.

On the first visit the screen will have been cleared and the message "KEYS ON" is printed on the top line. The usual Break information will then follow.

Since there has to be some way of cancelling the interrupt code, the program checks for the Shift-Lock key being held down as the Break key is pressed. If so the message "KEYS OFF" is printed instead, and the JMP at &287 is destroyed, preventing the second execution of the Break code.

On the second visit the interrupt is enabled once again, seemingly unaffected even by Ctrl-Break.

3. The interrupt code.

Two addresses hold information about what keys are currently held down:

&EC the last pressed

&ED the first (if more than one pressed)

In this program, the sequence Shift-Lock, then Caps-Lock will be used,

A Timed Interrupt Routine

which makes `?&EC=&C0` and `?&ED=&D0`. Should you want to use other keys, the simplest way is to run the one-liner:

```
REP. P. CHR$11;~?&EC,~?&ED:U. 0
```

pressing Escape when you have found what you want.

The entry states of the registers have to be preserved on the stack so that they can be restored at the end. It is also important that when the correct keys are pressed, the interrupt is disabled while the target code is obeyed and re-enabled afterwards.

4. The target code.

The code to be obeyed in response to correct key presses could be some code appended to this program, some other machine code in memory, or an operating system command. Obvious examples are a call to a screen dump routine or something like:

```
*save screen 5000 +3000
```

to save the current screen.

Here, the command:

```
*help
```

will be obeyed (see line 1830).

TESTING THE PROGRAM

Type in the listing carefully. Anything after line 1000 starting with a “\” may be omitted as these are only explanatory remarks. Save it BEFORE testing it as errors could corrupt the program itself.

Run the program. It should announce “KEYS ON” and repeatedly print out the alphabet. Interrupt it at any stage by holding Shift-Lock and pressing Caps-Lock. This should produce a “beep”, the usual *HELP messages, and then continue with the alphabet.

Press Break (even Ctrl-Break) and you should see “KEYS ON” at the top of the screen. Hold Shift-Lock and press Break to get the “KEYS OFF” message. Ctrl-Break will then reset the machine.

The program is now working properly and you can do something more useful with it! Change line 1830 to provide a more useful star command, or replace lines 1770-1840 with a different subroutine or a JMP to a subroutine at another address.

Do not run the program a second time without first disabling the interrupt. Those who do will find the machine very confused and will need to use the disabling technique to recover!

MEMORY PROBLEMS

Even though the demonstration program may work fine there can be problems when used with a real application.

For example, a printer dump routine run by including *pdump will load code into the computer's memory. This should not overwrite the code of this program. If it does, you should at least get one printout but the machine is then likely to “crash”. You need to discover where your ‘pdump’ “lives”. Enter:

```
*I.pdump
```

to get possibly something like:

```
pdump 900 12A 900
```

which tells you it loads at &900 and extends to &900 + &12A = &A2A. If you keep well clear of this by moving the interrupt code using, for example:

```
1080 P%=&A40
```

all should be well. Trial and error may be needed.

Other problems may arise because the main program may itself use the memory needed by the interrupt code, or by the target code. Experimenting may be the only answer!

```
10 REM Program INTERRUPT
20 REM Version 1.0
30 REM Author Robin Murphy
40 REM BEEBUG November 1991
50 REM Program subject to copyright
60 :
100 MODE 3:0%=0
```

A Timed Interrupt Routine

```
110 PROCassemble:CALL start
120 REPEAT
130 FOR I=1 TO 26:VDU64+I:NEXT
140 VDU 32
150 UNTIL 0
160 END
170 :
1000 DEFPROCassemble
1010 key1=&D0:REM first key held (SHIFT
LOCK)
1020 key2=&C0:REM last key held (CAPS L
OCK)
1030 key3=&D0:REM disable key (SHIFT LO
CK)
1040 time=&FFFF:REM to load Timer 1
1050 osascii=&FFE3:osword=&FFF1
1060 osbyte=&FFF4:cli   =&FFF7
1070 FOR opt=0 TO 3 STEP 2
1080 P%=&A00
1090 [OPT opt
1100 .start
1110 :
1120 \ set up interrupt
1130 SEI:JSR enable:CLI
1140 \ set BREAK vector
1150 LDA #&4C \ code for JMP
1160 STA &287
1170 LDA#break MOD 256:STA &288
1180 LDA#break DIV 256:STA &289
1190 .print_ON
1200 LDY #&FF
1210 .printloop
1220 INY:LDA on,Y:JSR osascii
1230 CMP #13:BNE printloop
1240 RTS
1250 .on
1260 EQU$ "KEYS ON"
1270 EQU$ 13
1280 .off
1290 EQU$ "KEYS OFF"
1300 EQU$ 13
1310 :
1320 .break
1330 BCS enable \ second visit
1340 \ is cancel key held?
1350 LDX #key3:LDA #121
1360 JSR osbyte
1370 TXA
1380 BPL print_ON
1390 \ disable BREAK vector
```

```
1400 INC &287
1410 LDY #off-on-1 \ to print "OFF"
1420 BNE printloop
1430 :
1440 .enable
1450 \ set IRQ2 vector
1460 LDA #interrupt MOD 256:STA &206
1470 LDA #interrupt DIV 256:STA &207
1480 \ set Timer1
1490 LDA #time MOD 256:STA &FE64
1500 LDA #time DIV 256:STA &FE65
1510 \ enable only Timer1 interrupts
1520 LDA #&3F:STA &FE6E
1530 .link
1540 LDA #&C0:STA &FE6E
1550 \ continuous interrupts
1560 STA &FE6B
1570 \ clear Timer1 flag
1580 STA &FE6D:RTS
1590 :
1600 .interrupt
1610 PHA:TXA:PHA:TYA:PHA
1620 LDA &ED
1630 CMP #key1:BNE continue
1640 LDA &EC
1650 CMP #key2:BNE continue
1660 \ correct keys pressed
1670 \ disable all VIA interrupts
1680 LDA#&7F:STA &FE6E
1690 JSR target_code
1700 .continue
1710 \ enable Timer1 interrupts
1720 JSR link
1730 PLA:TAY:PLA:TAX:PLA
1740 RTI
1750 :
1760 .target_code
1770 \ "beep"
1780 LDA #7:JSR osascii
1790 LDX #command MOD 256
1800 LDY #command DIV 256
1810 JMP cli
1820 .command
1830 EQU$ "HELP"
1840 EQU$ 13
1850 .finish
1860 ]:NEXT
1870 PRINT"ASSEMBLED FROM "~start" TO "
~finish
1880 ENDPROC
```

B

STEALTH

*A puzzle-game to exercise the right-hand side of your brain
by Patrick Dowling.*

'Stealth' attack force, invisible to radar, has to be pin-pointed for the missile batteries and you, the Array Laser Location Operator (ALLO!) are vital to the defence! You play against an opponent (or the computer), who sets a number of targets for you to find, and your success is rated for accuracy and the number of tries needed.

Enter the program carefully (Master128 owners can omit lines 0 & 1), and then save it.

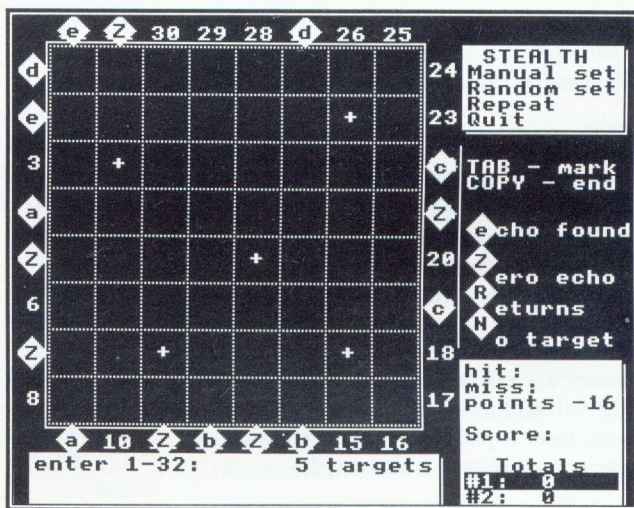
PLAY

Run the program and select *Manual* to move the cursor around with the cursor keys. Press Tab to set 4 or 5 targets - then press Copy to hide them for your opponent to find. Alternatively, select *Random* and let the computer set them for you.

Enter numbers between 1 and 32, and diamond shapes will show where your laser enters and leaves the grid. Its path will be influenced by any targets it meets, and you have to work out where they must be. Move the cursor to where you think they are and mark them, or erase a previously set mark, by pressing Tab.

Stealth targets give no response to a direct hit, but a laser touching a corner

of a target is bounced away at right-angles, and this may happen more than once.



A game in progress

A RED diamond means Zero Echo, indicating that a target must have taken a direct hit at some point in the laser's travel.

YELLOW diamonds mark entry and exit points, indicating that the laser has bounced off a target one or more times.

A WHITE diamond (lettered "R") means Return - a laser has been bounced back to its point of entry because it met targets on both sides of its path giving it a double bounce, or a target lies on the edge of the grid next to the point of entry.

WHITE diamonds (lettered "N") mean No Target found on, or to either side of a direct path.

STEALTH

It will take several tries to locate all of them, but when all targets are marked press Copy to find out how successful you have been.

Lasers only travel vertically and horizontally, never diagonally. Experiment with targets in known positions to see the effect. You will find that even if a target cannot be touched, its position can always be deduced.

SCORING

Two scores are kept. Players take turns, the winner being the one with the most points over 50. A good player would need 5 games to reach this total. Reaching it in 4 would be excellent!

The score for each hit depends on how many are set (the more targets the less the value,) but each diamond around the grid loses a point, and also the value of each target missed is deducted.

This month's magazine disc has an extended version of the game including a demonstration, and a facility to store favourite set-ups.

```
0 *KEYO VDU21|M*TAPE|MFORN%=0TOTOP-P
AGE STEP4:N%!&E00=N%!PAGE:NEXT|MPAGE=&E0
0|MOLD|MRUN|M
1 IFFPAGE>&E00 AND HIMEM<&8000 VDU21:
*FX138,0,128
10 REM Program Stealth
20 REM Version B.10M (mag version)
30 REM Author Patrick Dowling
40 REM BEEBUG November 1991
50 REM Program subject to copyright
60 :
100 VDU6:MODE1
110 ONERROR PROCend:END
120 PROCinit:pp=1
130 PROCbox(2):PROCbox(3)
```

```
140 REPEAT
150 PROCbox(1):PROCset
160 IF pp<>4 PROCsolve
170 UNTIL pp=4
180 PROCend:CLS:END
190 :
1000 DEF PROCbox(j%)
1010 RESTORE 1140
1020 FOR k%=1 TO j%
1030 READ X%,lines,wid,Y%
1040 NEXT
1050 VDU26:PROCcurs(0)
1060 VDU29,X%*32-8;1024-(Y%+lines)*32;
1070 xx%=wid*32+16:yy%=lines*32+4
1080 GCOL0,2:MOVE8,-12
1090 DRAW xx%+8,-12:DRAW xx%+8,yy%-8
1100 GCOL0,131:VDU24,0;-4;xx%;yy%;16
1110 VDU28,X%,lines+Y%,X%+wid-1,Y%
1120 PROCmenu(j%)
1130 ENDPROC
1140 DATA9,5,10,2
1150 DATA1,3,26,28
1160 DATA29,9,10,22
1170 DATA29,8,10,11
1180 :
1190 DEF PROCmenu(j%)
1200 IF j%=1 RESTORE 1300
1210 IF j%=2 RESTORE 1310
1220 IF j%=3 RESTORE 1320
1230 COLOUR0:COLOUR131
1240 FOR K%=0 TO lines-1
1250 READ V$:PRINTTAB(0,K%)V$
1260 NEXT
1270 IF j%=3 FOR k%=0 TO 1:PRINT TAB(5,
7+k%);tot(k%);:NEXT
1280 PROCnoky
1290 ENDPROC
1300 DATA " STEALTH",Manual set,Random
set,Repeat,Quit
1310 DATA enter 1-32:,,
1320 DATA hit:,miss:,points:,Score:,,
Totals",#1:,#2:
1330 :
1340 DEF PROCset
1350 REPEAT
```

```

1360 pp=FNbarmove
1370 IF G%=27 pp=4
1380 UNTIL G%=13
1390 IF pp=4 ENDPROC
1400 IF pp<3 FOR k%=0TO64:grid%?k%=k%:N
EXT
1410 IF pp<3 FOR k%=1TO8:targ%?k%=0:NEX
T
1420 PROCbox(3)
1430 T%=T% EOR1:PROcbar(T%+7)
1440 IF pp=1 PROCmanual
1450 IF pp=2 PROCrandom
1460 IF pp=3:FOR k%=0TO63:grid%?k%=grid
%?k% AND191:NEXT
1470 ENDPROC
1480 :
1490 DEF PROCmanual
1500 VDU26
1510 PRINTTAB(1,28)"enter 0-63:"SPC15 T
AB(1,30)"Set 3-7 targets"SPC11;
1520 PROCgrid:PROccross
1530 REPEAT:PROccurs(0)
1540 REPEAT:j%=FNinput
1550 UNTIL j%<64
1560 Q%=LEN V$
1570 IF G%>135 PROCmoveX:GOTO 1600
1580 IF G%=9 j%=cx%+cy%*8 ELSE IF Q% PR
OCcross:cx%=j% MOD8:cy%=j% DIV8:PROccros
s
1590 IF G%=9 OR Q%>0 PROCmark(1,1):grid
%?j%=grid%?j% EOR128
1600 UNTIL G%=135 OR G%=27
1610 P%=0
1620 FOR j%=0 TO 63
1630 IF(grid%?j% AND 128) AND P%<7 P%=P
%+1:targ%?P%=j% ELSE grid%?j%=j%
1640 NEXT
1650 ENDPROC
1660 :
1670 DEF PROCrandom
1680 P%=RND(-TIME)MOD2 +5
1690 FOR K%=1 TO P%
1700 REPEAT
1710 X%=RND(8)-1:IF X%=0 OR X%=7 X%=RND

```

```

(8)-1
1720 Y%=RND(8)-1:IF Y%=0 OR Y%=7 Y%=RND
(8)-1
1730 j%=X%+Y%*8:Q%=TRUE
1740 FOR k%=1 TO K%
1750 IF targ%?k%=j% Q%=0
1760 NEXT
1770 UNTILQ%
1780 targ%?K%=j%:grid%?j%=j%+128
1790 NEXT
1800 ENDPROC
1810 :
1820 DEF PROCsolve
1830 IF P%=0 VDU7:ENDPROC
1840 PROCgrid:PROccross
1850 char=ASC"a":S%=0:H%=0:M%=0
1860 PROCbox(2)
1870 PRINT TAB(17,0) SPC10 TAB(17,0);P%
" targets"
1880 REPEAT:PROctry
1890 COLOUR0:COLOUR131
1900 VDU28,29,31,39,22:PRINTTAB(7,2);S%
1910 UNTIL FNfinish
1920 ENDPROC
1930 :
1940 DEF PROctry
1950 I%=FNinput:PRINTTAB(0,2) SPC26;
1960 IF G%=9 VDU26:PROCmark(0,1):j%=cx%
+cy%*8:grid%?j%=grid%?j% EOR64:ENDPROC
1970 IF G%>135 VDU26:PROCmoveX:ENDPROC
1980 IF G%=135 OR G%=27:ENDPROC
1990 IF I%<1 OR I%>32:ENDPROC
2000 X%=0:Y%=I%-1:step%=1
2010 IF I%>8 X%=I%-9:Y%=7:step%=-8
2020 IF I%>16 X%=7:Y%=24-I%:step%=-1
2030 IF I%>24 X%=32-I%:Y%=0:step%=8
2040 pos%=X%+Y%*8
2050 side%=(I%-1)DIV8:in=side%
2060 PROCpath:PROccause:PROceffect
2070 ENDPROC
2080 :
2090 DEF PROCpath
2100 E%=pos% MOD8
2110 FOR B%=0TO7:FOR j%=0TO2

```

STEALTH

```
2120 IF side%<2 J%=j%-1 ELSE J%=1-j%
2130 K%=pos%+B%*step%+J%*(ABSstep% EOR9
)
2140 IF K%<0 OR K%>63 K%=64
2150 IF side%=1:IF ((j%=0 AND E%=0) OR
(j%=2 AND E%=7)) K%=64
2160 IF side%=3:IF ((j%=0 AND E%=7) OR
(j%=2 AND E%=0)) K%=64
2170 path%?(B%+j%*8)=grid%?K%
2180 NEXT:NEXT
2190 ENDPROC
2200 :
2210 DEF PROCcouse
2220 IF path%?8 >127 A%=4:ENDPROC:ELSE
IF ?path%>127 OR path%?16>127 A%=3:ENDPR
OC
2230 defl=0:B%=-1
2240 REPEAT:A%=0
2250 REPEAT
2260 IF path%?(B%+1) >127 A%=1
2270 IF path%?(B%+17) >127 A%=A% OR2
2280 IF path%?(B%+9) >127 A%=4
2290 IF A%=0 AND B%<7 B%=B%+1
2300 UNTIL A%>0 OR B%=7
2310 IF A%=1 OR A%=2 PROCdefl:UNTIL B%=
7 ELSE UNTIL TRUE
2320 ENDPROC
2330 :
2340 DEF PROCdefl
2350 IF A%=1 side%=side%-1 ELSE side%=s
ide%+1
2360 IF side%<0 side%=3 ELSE IF side%>3
side%=0
2370 pivot%=path%?(B%+8):defl=TRUE
2380 IF side%=0 pos%=pivot% DIV8*8:step
%=1
2390 IF side%=1 pos%=56+pivot% MOD8:ste
p%=-8
2400 IF side%=2 pos%=pivot% DIV8*8+7:st
ep%=-1
2410 IF side%=3 pos%=pivot% MOD8:step%=
8
2420 PROCpath:B%=-1
2430 REPEAT
```

```
2440 B%=B%+1
2450 UNTIL path%?(B%+8)=pivot%
2460 ENDPROC
2470 :
2480 DEF PROCeffect
2490 xx%=path%?15 MOD8:yy%=path%?15 DIV
8
2500 IF side%=0 exit=24-yy%
2510 IF side%=1 exit=32-xx%
2520 IF side%=2 exit=yy%+1
2530 IF side%=3 exit=9+xx%
2540 IF A%=4 defl=0:V$=" ZERO ECHO":S%=
S%-1:out=in:char$="Z":col=1
2550 IF A%=3 defl=0:V$=" RETURNS ":S%=S
%-1:out=in:char$="R":col=3
2560 IF defl V$="ECHO found at "+STR$ex
it:S%=S%-2:out=(side%+2)MOD4:char$=CHR$c
har:col=2
2570 IF defl=0 AND A%=0 V$=" NO TARGET
to "+STR$exit:S%=S%-2:out=(in+2)MOD4:cha
r$="N":col=3
2580 PRINTTAB(0,2)SPC26;TAB(0,2)"Laser
";I%;TAB(9,2)"-V$;
2590 PROCdiamond(in,64+X%*96,960-Y%*96)
2600 IF A%<3 PROCdiamond(out,64+xx%*96,
960-yy%*96):IF defl char=char+1
2610 ENDPROC
2620 :
2630 DEF PROCdiamond(j%,X%,Y%)
2640 VDU5,26:GCOL0,col
2650 IF j%=0 MOVE X%-68,Y%-48
2660 IF j%=1 MOVE X%+16,Y%-128
2670 IF j%=2 MOVE X%+108,Y%-48
2680 IF j%=3 MOVE X%+16,Y%+32
2690 PLOT80,32,32:PLOT81,0,-64
2700 PLOT81,32,32
2710 GCOL0,0:PLOT0,-48,12:PRINTchar$
2720 VDU4
2730 ENDPROC
2740 :
2750 DEF FNfinish
2760 IF G%=27 =TRUE ELSE IF G%<135 =FA
LSE
2770 PROCcross:j%=0
```

```

2780 FOR k%=0 TO 63
2790 IF grid%?k% AND 64 j%=j%+1
2800 NEXT
2810 VDU28,1,31,32,28
2820 IF j%=P% PRINT TAB(0,2) SPC26:PROC
score:=TRUE
2830 IF j%<P% V$="few" ELSE V$="many"
2840 COLOUR1:COLOUR131:PROCsound(2)
2850 PRINTTAB(12,2)"Too "V$" marks";
2860 COLOUR0:PROCcross
2870 =FALSE
2880 :
2890 DEF PROCscore
2900 PROCnoky:VDU28,29,31,39,22
2910 FOR k%=0 TO 63
2920 cx%=k% MOD8:cy%=k% DIV8
2930 K%=grid%?k%
2940 IF K% AND 128 PROCsound(0):IF K% A
ND 64 PROCsound(1):H%=H%+10-P%:PRINTTAB(
7,0);H%
2950 IF K%AND 128 PROCmark(1,5):IF INKE
Y25 ELSE IF K% AND 64 PROCsound(2):PROCM
ark(0,4):M%=M%-10+P%:PRINTTAB(7,1);M%:IF
INKEY25
2960 NEXT
2970 PRINTTAB(7,0);H% TAB(7,1)M% TAB(7,
2);S%
2980 S%=S%+H%+M%:PRINTTAB(7,4);S%
2990 tot(T%)=tot(T%)+S%:IF tot(T%)<0 to
t(T%)=0
3000 COLOUR3:COLOUR128
3010 PRINT TAB(5,7+T%) SPC4;TAB(5,7+T%)
;tot(T%)
3020 VDU26:COLOUR0:COLOUR131
3030 PRINT TAB(23-P%*3,30)"at:";
3040 FOR k%=1 TO P%
3050 PRINTTAB(POS,30)" ";targ%?k%;
3060 NEXT
3070 ENDPROC
3080 :
3090 DEF FNbarmove
3100 REPEAT
3110 PROCnoky:PROCbar(pp)
3120 G%=GET:PROCbar(pp)

```

```

3130 IF G%=138 pp=pp+1 ELSE IF G%=139 p
p=pp-1
3140 IF pp<1 pp=lines-1 ELSE IF pp>line
s-1 pp=1
3150 UNTIL G%<138
3160 PROCnoky
3170 =pp
3180 :
3190 DEF PROCbar(pp)
3200 VDU24,8;(lines-pp-1)*32;wid*32+8;(
lines-pp)*32;
3210 GCOL4,128:CLG
3220 ENDPROC
3230 :
3240 DEF FNinput
3250 PROCcurs(1):V$=""
3260 COLOUR0:COLOUR131:VDU28,1,31,32,28
3270 REPEAT
3280 PRINT TAB(12,0) SPC2 TAB(12,0)V$;
3290 PROCnoky:G%=GET
3300 IF G%=127 V$=LEFT$(V$,LENV$-1)
3310 IF G%=32 OR G%=9 V$=""
3320 IF G%<58 AND G%>47 V$=V$+CHR$G%:V$
=LEFT$(V$,2)
3330 UNTIL G%<32 OR G%>127
3340 PROCcurs(0)
3350 =VAL V$
3360 :
3370 DEF PROCmoveX
3380 PROCcross
3390 IF G%=136 cx%=cx%-1 ELSE IF G%=137
cx%=cx%+1
3400 IF G%=138 cy%=cy%+1 ELSE IF G%=139
cy%=cy%-1
3410 IF cx%<0 cx%=7 ELSE IF cx%>7 cx%=0
3420 IF cy%<0 cy%=7 ELSE IF cy%>7 cy%=0
3430 PROCcross
3440 ENDPROC
3450 :
3460 DEF PROCcross
3470 GCOL3,3:MOVE cx%*96+64+16,864-cy%*
96+80
3480 PLOT17,64,-64:PLOT16,-64,0
3490 PLOT17,64,64:GCOL0,3

```

STEALTH

```
3500 ENDPROC
3510 :
3520 DEF PROCmark(j%,J%)
3530 GCOL3,3:VDU5
3540 FOR Q%=1 TO J%
3550 MOVE cx%*96+64+32,864-cy%*96+64
3560 PRINT CHR$(224+j%):IF INKEY 10
3570 NEXT
3580 GCOL0,3:VDU4
3590 ENDPROC
3600 :
3610 DEF PROCnoky
3620 REPEAT
3630 UNTIL INKEY(-129)
3640 *FX15,1
3650 ENDPROC
3660 :
3670 DEF PROCcurs(j%)
3680 VDU23,1,j%;0;0;0;
3690 ENDPROC
3700 :
3710 DEF PROCsound(j%)
3720 IF j%=0 SOUND1,-10,136,5
3730 IF j%=1 SOUND1,-12,120,5
3740 IF j%=2 SOUND1,-12,36,5
3750 ENDPROC
3760 :
3770 DEF PROCinit
3780 DIM tot(1),path% 24,targ% 8,grid%
64
3790 P%=0:T%=1:S%=0
3800 VDU23,224,0,24,24,126,24,24,0,0
3810 VDU23,225,24,126,126,255,255,126,1
26,24
3820 OSCLI"FX220":OSCLI"FX4,1"
3830 VDU19,0,4;0;:PROCgrid:VDU26
3840 MOVE 918,760:DRAW 918,352
3850 VDU28,29,22,39,9
3860 PRINT"TAB - mark""COPY - end"
3870 PRINT"" cho found"" ero echo"
"" eturns"" o target";
3880 RESTORE 3930
3890 FOR Y%=560 TO 368 STEP-64
3900 READ col,char$:PROCdiamond(3,916,Y
```

```
%)
3910 NEXT
3920 ENDPROC
3930 DATA 2,e,1,Z,3,R,3,N
3940 :
3950 DEF PROCgrid
3960 VDU26,24,0;136;914;1020;
3970 GCOL0,2:GCOL0,128:CLG
3980 MOVE 840,968
3990 DRAW 840,184:DRAW 56,184
4000 DRAW 56,968:DRAW 840,968
4010 GCOL0,3:VDU5
4020 FOR X%=64 TO 832 STEP96
4030 PLOT 20,X%,192:PLOT 21,X%,960
4040 NEXT
4050 FOR Y%=192 TO 960 STEP96
4060 PLOT 20,64,Y%:PLOT 21,832,Y%
4070 NEXT
4080 N%=1
4090 FOR Y%=928 TO 256 STEP-96
4100 MOVE 16,Y%:PRINT;N%
4110 N%=N%+1
4120 NEXT
4130 MOVE 92,168:PRINT;N%
4140 N%=N%+1
4150 FOR X%=176 TO 752 STEP96
4160 MOVE X%,168:PRINT;N%
4170 N%=N%+1
4180 NEXT
4190 FOR Y%=256 TO 928 STEP96
4200 MOVE 848,Y%:PRINT;N%
4210 N%=N%+1
4220 NEXT
4230 FOR X%=752 TO 80 STEP-96
4240 MOVE X%,1000:PRINT;N%
4250 N%=N%+1
4260 NEXT:VDU4:cx%=4:cy%=4
4270 ENDPROC
4280 :
4290 DEF PROCend
4300 REPORT:PRINT" @ ";ERL:ONERROR OFF
4310 OSCLI"FX220,27":OSCLI"FX4"
4320 VDU3,4,20,26
4330 ENDPROC
```



Wordwise User's Notebook

Wordwise Plus Teletext Effects (2)

Chris Robbins continues his exploration of teletext effects using Wordwise Plus.

Yet another use of colour is to 'improve' the background to text and/or graphics. One method of applying a background 'wash' is to 'paint' successive coloured lines across the screen using the graphics character with ASCII code 255 as a brush. In the following example blue 'paint' has been used:

```
CLS
VDU23;11,0;0;0;0;
L%=0
REPEAT
REM Select line to be 'painted'
  VDU31,0,L%
REM Select blue 'paint'
  VDU148
  REPEAT
REM Apply paint
  VDU255
  TIMES 39
  L%=L%+1
UNTIL L%>23
VDU31,0,12
VDU131,157,132
PRINT "Hello";
VDU131,157
```

The end result is my contribution to the modern school of ultra-simplistic painting, which I've provisionally entitled "Blue Yellow Greeting on Blue". But of course, one needn't use this technique for such simple artistic ends. It also happens to be fairly slow. A considerably faster and simpler way to slap the paint on is:

```
CLS
VDU23;11,0;0;0;0;0;
```

```
REPEAT
REM Select blue background
VDU132,157
REM Paint a blue line
PRINT
TIMES 24
```

Against such a colourful background, the relevant prompts, text, user input or whatever, can be displayed to maximum effect. For example, the following could be appended to the preceding to give the beginnings of a WW+ on-line Help feature:

```
VDU31,2,1
REM Use yellow text
VDU131
PRINT "CTRL-A Delete At cursor"
VDU31,2,2
VDU131
PRINT "CTRL-S Change Case"
VDU31,2,3
VDU131
PRINT "CTRL-D Delete Word"
VDU31,2,4
VDU131
PRINT "CTRL-" + CHR$(93) +
  " Find Next Word"
VDU31,2,5
VDU131
PRINT "CTRL-" + CHR$(91) +
  " Find Previous Word"
VDU31,2,6
VDU131
PRINT "SHIFT-" + CHR$(93) +
  " Find End of Line"
VDU31,2,7
VDU131
```

Wordwise User's Notebook

```
PRINT "SHIFT-"+CHR$(91)+  
  " Find Start of Line"
```

Pressing Shift and the appropriate function key will bring up the Help screen. Possible enhancements could include highlighting key words in the display, inclusion of *all* control key operations, embedded commands, multiple or separate pages for different topics, etc.

Finally, to make things even more interesting, areas of the screen could be painted to provide different backgrounds (for different key groups say). For example:

```
CLS  
VDU23;11,0;0;0;0;  
  
REM Set blue background  
REPEAT  
  VDU132,157  
REM Paint a blue line  
  PRINT
```

```
TIMES 6  
  
REM Set yellow background  
REPEAT  
  VDU131,157  
  PRINT  
TIMES 6  
  
REM Set red background  
REPEAT  
  VDU129,157  
  PRINT  
TIMES 6  
  
REM Set magenta background  
REPEAT  
  VDU133,157  
  PRINT  
TIMES 6
```

Hopefully, the ideas discussed here and in part 1 will give you plenty of food for thought. And if you come up with any original ideas of your own why not send them in to the editor?



Points Arising...Points Arising...Points Arising...Points Arising...

FUNCTION/PROCEDURE LIBRARY (Vol.9 No.9)

The BEEBUG Function/Procedure library has proved very popular since we started in Vol.9 No.9. However, it has come to our attention, courtesy Mr.N.P.Fay, that there were some inaccuracies, and indeed a couple of bugs in some of the earliest routines.

Routine 3: the parameters X2% and Y2% are lengths, and not co-ordinates as stated.

Routine 4: the function FNcut returns the offset needed to centre a string - it

does not centre the string itself. In line 20350, a figure zero (0) was incorrectly (but obviously) misprinted as a letter 'oh' (O).

Routine 7: Line 20560 should read:
20560 VDU23,16,M%*2;0;0;0;

Routine 12: The list of related functions does not include FNswap which is called by the related FNstyle.

Our apologies if these mistakes caused any confusion.



TexBase (3)

by Paul Pibworth

In the last two articles you learned how to wet up and use the text database, TexBase. This month there is a short listing (listing 1) called *Setup*. This reads the free space on a disc, and then creates the data file text for you. If you type in this program, I would suggest that you keep it on a separate disc, as a TexBase utility.

```
ENTER THE PAGE TO BE DELETED 1
TexBase Part 3

The process works in mode 7, to make mor
e use of memory.
It is necessary to know the page number
to be deleted,
as there is no keyword search facility.

The first few lines of the page are disp
layed for confirmation.
However, because mode 7 uses a 40 charac
ter screen,
some words will be split - this is not i
mportant.

IS THIS THE PAGE?
```

Deleting a page with TexBase

This month's main program is listing 2. It is called *Delete*, and its purpose is to remove one page (at a time) from the data file. It is similar in concept to a *COMPACT on a disc, in that all the rest of the data is shunted down inside the file text after the deletion (remember, text is a large random access file, initially empty, which is used to store the entries as "pages").

You will recall that new pages in *text* begin at 77 byte intervals. If a page is deleted, it can be regarded as a gap within *text*, which must be filled. A block of data from the end of this gap is read into a buffer, and then saved back to the disc at a lower address on the disc, thus filling the gap. This creates a new gap, so the process is repeated until the end of the file

is reached, or the last page within the file is reached. I suggest making a backup of the data first, in case something goes wrong. If the file is quite full, and you have deleted an early page, then it will take some time. This is because each page has its page number stored along with its text. Each page has to be renumbered, and the index has to be modified.

The process works in mode 7, to make more use of memory. It is necessary to know the page number to be deleted, as there is no keyword search facility. The first few lines of the page are displayed for confirmation. However, because mode 7 uses a 40 character screen, some words will be split - this is not important.

As before, it would be wise to replace *CHAIN"TMENU"* with *END*, and check the program works on a test version of text. Then replace the *CHAIN"TMENU"*. You could also have a larger buffer at line 120, e.g. *DIM store% 10000*, in which case you should change the 4999 at lines 1430 and 1440 to 9999.

In the first article, you may remember that it was possible to edit a page, but only within certain constraints: that the edited version took up the same number of lines, i.e. filled the same gap within "text". What if there is a need to extend? In this case, the page must be saved, as described last month, possibly on another disc. Then the original page must be deleted, as described above. Finally, the new page can be reloaded from the other disc using *Load ASCII file*, edited, and saved. It would be necessary to note the title and keywords, as these are not saved with the text.

So there it is, a text database. As I write this, further suggestions come to mind. There could be a utility to print out all the page numbers, titles, and keywords. Another possibility is a program to copy a page or pages from text on one disc to text on another disc. And if a page can be deleted, can one be inserted? There may be other ideas. There is one final good point: if you upgrade to an Archimedes, the system still works!

Listing 1

```
10 REM Program Setup
20 REM Version B1.0
30 REM Author Paul Pibworth
40 REM BEEBUG November 1991
50 REM Program Subject to Copyright
60 :
100 ON ERROR GOTO 330
110 PROCgetcat
120 tr%=(?&906 AND 3)*256+?&907
130 C%=?&90C:D%=?&90D
140 E%=?&90E:F%=?&90F
150 bit45%=E% AND 48:bit45%=bit45%*16
160 bit01%=E% AND 3:bit01%=bit01%*256
170 len%=D%+bit45%:IF C%>1 len%=len%+1
180 lf%=bit01%+F%
190 PRINT"No of tracks=";tr%/10
200 free%=(tr%-lf%-len%)*256
210 PRINT"Free space=";free%
220 lines%=(free%-&200)/DIV 77
230 PRINT"No of lines=";lines%
240 Z%=OPENIN"TEXT":IF Z%>0 CLOSE#Z%:P
RINT"THE DATA FILE IS ON THIS DISC!":END
250 INPUT"Enter size of data file (in
lines):"size%
260 IF size%>lines% size%=lines%
270 IF size%<50 size%=50
280 size%=size%*77+&200
290 Z%=OPENOUT("text")
300 FOR I%=1 TO size%
310 BPUT#Z%,0
320 NEXT
330 CLOSE#Z%
340 END
350 :
1000 DEFPROCgetcat
1010 ?&80=0 :REM drive
```

```
1020 !&81=&900:REM destination
1030 ?&85=&3 :REM 3 parameters
1040 ?&86=&53 :REM read
1050 ?&87=0 :REM track 0
1060 ?&88=1 :REM sector 1
1070 ?&89=&21 :REM look at 1 sector
1080 X%=&80:Y%=0:A%=&7F:CALL&FFF1
1090 ENDPROC
```

Listing 2

```
10 REM Program Delete
20 REM Version B1.0
30 REM Author Paul Pibworth
40 REM BEEBUG November 1991
50 REM Program Subject To Copyright
60 :
100 MODE 7
110 CLOSE#0
120 DIM A$(6),index% 512,store% 5000
130 ON ERROR GOTO2020
140 pr$="PRESS return"
150 Z%=OPENUP("text")
160 ext%=EXT#Z%
170 FOR I%=0 TO 511
180 index%?I%=BGET#Z%
190 NEXT
200 CLOSE#Z%
210 REPEAT:PROCmain:UNTIL M%=50
220 CLOSE#0
230 CHAIN"TMENU"
240 :
1000 DEFPROCmain
1010 CLS:VDU28,10,15,39,5
1020 PRINT"1...DELETE A PAGE""2...END"
1030 REPEAT:M%=GET:UNTIL M%=49 OR M%=50
1040 IF M%=49 VDU26:PROCcont
1050 CLOSE#0
1060 ENDPROC
1070 :
1080 DEFNpointer(p%)
1090 IF p%=1 THEN =&200
1100 byt%=p%*2-3
1110 ptr%=index%?byt%
1120 ptr%=ptr%*256+index%?(byt%+1)
1130 =&200+77*ptr%
1140 :
1150 DEFPROCcont
1160 CLS:PRINTTAB(5,5)"ENTER THE PAGE T
O BE DELETED ";
```

```

1170 REPEAT:INPUT"fp%:UNTIL fp%>0 AND
fp%<256
1180 G%=?index%
1190 IF fp%>G% PRINT'CHR$132;"NO SUCH P
AGE!":A%=INKEY(200):ENDPROC
1200 P%=FNpointer(fp%)
1210 Z%=OPENUP("text")
1220 PTR#Z%=P%:ptr1%=P%:start%=P%
1230 INPUT#Z%,pg%,lines%,kw$,ti$
1240 P%=P%+154:PTR#Z%=P%
1250 N%=1:REPEAT
1260 INPUT#Z%,A$(N%)
1270 N%=N%+1
1280 UNTIL N%=7 OR N%>lines%
1290 PRINTti$:PRINT
1300 FOR N%=1 TO 6
1310 PRINTA$(N%)
1320 NEXT
1330 PRINT'CHR$131"IS THIS THE PAGE?"
1340 REPEAT:R$=CHR$(GETAND95):UNTIL INS
TR("YN",R$)>0
1350 CLS
1360 IFR$<>"Y" ENDPROC
1370 PRINTTAB(5,10);CHR$133;"PLEASE WAI
T, COMPACTING"
1380 P%=ptr1%
1390 ptr2%=FNpointer(fp%+1)
1400 IF fp%=G%:PROClast:PROCindex:ENDPR
OC
1410 end%=FNpointer(G%)
1420 REPEAT
1430 loop%=4999
1440 IF ext%-ptr2%<4999 loop%=ext%-ptr2
%
1450 PTR#Z%=ptr2%
1460 FOR I%=0 TO loop%
1470 store%?I%=BGET#Z%
1480 NEXT
1490 ptr2%=PTR#Z%
1500 PTR#Z%=ptr1%
1510 FOR I%=0 TO loop%
1520 BPUT#Z%,store%?I%
1530 NEXT
1540 ptr1%=PTR#Z%:PRINT
1550 UNTIL ptr2%=ext% OR ptr1%>end%
1560 PROCrepage:PROCindex
1570 ENDPROC
1580 :

```

```

1590 DEFPROClast
1600 PTR#Z%=ptr1%
1610 FOR I%=ptr1% TO FNpointer(G%+1)-1
1620 BPUT#Z%,0:NEXT
1630 byt%=(pg%+1)*2-3
1640 index%?byt%=0
1650 index%?(byt%+1)=0
1660 ENDPROC
1670 :
1680 DEFPROCrepage
1690 PRINTTAB(5,12);CHR$131"NOW RE-PAGI
NG..."
1700 P%=start%:REPEAT
1710 PTR#Z%=P%
1720 INPUT#Z%,pg%,sz%
1730 ptr%=(P%-&200)/77
1740 byt%=(pg%-1)*2-3
1750 index%?byt%=ptr% DIV 256
1760 index%?(byt%+1)=ptr% MOD 256
1770 PTR#Z%=P%
1780 PRINT#Z%,pg%-1
1790 P%=P%+154+77*sz%
1800 UNTIL pg%=G%
1810 ptr%=(P%-&200)/77
1820 byt%=(pg%)*2-3
1830 index%?byt%=ptr% DIV 256
1840 index%?(byt%+1)=ptr% MOD 256
1850 index%?(byt%+2)=0
1860 index%?(byt%+3)=0
1870 ENDPROC
1880 :
1890 DEFPROCindex
1900 ?index%=G%-1
1910 PTR#Z%=0
1920 FORI%=0 TO 511
1930 BPUT#Z%,(index%?I%)
1940 NEXT:CLOSE#Z%
1950 PRINTTAB(5,14)CHR$132"D O N E ! "
1960 a=INKEY(200):CLS
1970 ENDPROC
1980 :
1990 DEFPROCend:VDU14:CHAIN"TMENU":END
2000 :
2010 REM ERROR TRAP
2020 CLOSE#:REPORT
2030 PRINTERR;" at ";ERL:
2040 PRINTpr$:B%=GET:IF B%=13 GOTO210
2050 OSCLI"*FX229,0":END

```

B

Magscan

Comprehensive
Magazine Database
for the BBC Micro and
the Master 128

An updated version of Magscan, which contains the complete indexes to all BEEBUG magazines from
Volume 1 Issue 1 to Volume 10 Issue 5

Magscan allows you to locate instantly all references to any chosen subject mentioned anywhere in the 90 issues of BEEBUG magazine.

Just type in one or two descriptive words (using AND/OR logic), and you can find any article or program you need, together with a brief description and reference to the volume, issue and page numbers. You can also perform a search by article type and/or volume number.

The Magscan database can be easily updated to include future magazines. Annual updates are available from BEEBUG for existing Magscan users.

```
BEEBUG MAGSCAN          VOLUMES 1-9
Enter Volume No. (1-9 or *) > * <
Enter article type      > * <
# - All types
A - General Article
B - Programming Article
C - Review
D - News
E - Hint
F - Points Arising
G - Application Program
H - Utility Program
I - Games Program
J - Miscellaneous

Enter String 1          > Basic   <
Enter String 2          > Program <
Logic OR/AND (O/A)     > AND    <
Hard Copy (Y/N)        > N     <
```

Specifying a Magscan search

```
BEEBUG MAGSCAN          VOLUMES 1-9
Volume      : 1 2 3 4 5 6 7 8 9
Type        : All
String 1    : BASIC
String 2    : PROGRAM
Logic       : AND

Editit (Part 5)
Basic Program Utility/Toolkit ROM
Programming Utilities
Vol 9 No 1 Page 30

Thanks for the Memory - Bas128 (Part 1)
Main Memory Resident Version of Basic
Sideways RAM Program Storage
Vol 9 No 3 Page 20

Hint: Improved Move-Down Routine
Using Additional Program Lines
Basic/PAGE/Memory Restrictions
Vol 9 No 4 Page 61
```

Entries retrieved from Magscan files

Some of the features Magscan offers include:

- ◆ full access to all BEEBUG magazines
- ◆ rapid keyword search
- ◆ flexible search by volume number, article type and up to two keywords
- ◆ keyword entry with selectable AND/OR logic
- ◆ extensive on-screen help
- ◆ hard copy option
- ◆ easily updatable to include future magazines
- ◆ yearly updates available from BEEBUG

Phone your order now on (0727) 40303
or send your cheque/postal order to the address below. Please quote your name and membership number. When ordering by Access, Visa or Connect, please quote your card number and the expiry date.

Magscan complete pack, contains disc and revised manual: **£9.95+p&p**

Stock codes: 0005a 5.25"disc 40 track DFS 1457a 3.5" ADFS disc
0006a 5.25"disc 80 track DFS

Magscan update, contains disc and revised manual: **£4.75 +p&p**

(for update, please return old disc, label or evidence of purchase)

Stock codes: 0011a 5.25"disc 40 track DFS 1458a 3.5" ADFS disc
0010a 5.25"disc 80 track DFS

Please add p&p (for details see back cover)

Patterns from a Keyboard (continued from page 12)

```

2160 @%=&02010A
2170 PRINT c$"Length: ";LENA$" character
rs"
2180 PRINT y$"Unit is ";LENA$/p "" "" wid
e"
2190 ENDPROC
2200 :
2210 DEF PROCnextStage
2220 PROCparamMenu:PROCfixBlock
2230 PROCshape:PROCjustify
2240 IF text PROCjoinCrop
2250 IF NOT join PROCgradBandUni
2260 IF band PROCincrDecr
2270 IF unif PROCsetFeed
2280 IF shap PROCupDown
2290 IF rect OR join PROCdepth
2300 IF text AND NOT(join OR crpd OR un
Mo) AND NOT cent PROCoblqVert
2310 PROCtraverse
2320 PROCwnd(0,24,39,23,-1)
2330 PRINT c$"OK (Y/N)? ";
2340 IF FNip(1,-1,"YN")="N" CLS:PROCnex
tStage:ENDPROC
2350 IF join PROCjoin ELSE PROCprintBlo
ck
2360 REM IF NOT join PROCreverse
2370 REM IF NOT revr PROCrecord
2380 PROCfinish:ENDPROC
2390 :
2400 DEF PROCparamMenu
2410 PROCwnd(0,16,39,3,-1)
2420 RESTORE 2490:FOR a%=1 TO 13
2430 READ col%,opt$:PRINT CHR$col% opt$
;
2440 IF a% <14 PRINT
2450 NEXT:PROCwnd(0,16,39,15,0)
2460 PRINT c$ STRING$(38,"*");
2470 PROCwnd(0,16,39,3,0):ENDPROC
2480 :
2490 DATA 131,"Left margin (inches
from LHS)?"
2500 DATA 134,"Right margin (inches
from LHS)?"
2510 DATA 131,"Shaped or Rectangular
(S/R)?"
2520 DATA 134,"Justify-Left/Centre/Righ
t (L/C/R)?"

```

```

2530 DATA 131,"Joined/Cropped/Unmodifie
d (J/C/U)?"
2540 DATA 134,"Uniform or Banded
(U/B)?"
2550 DATA 131,"Number of bands ?"
2560 DATA 134,"Increase/Decrease feed
(I/D)?"
2570 DATA 131,"Enter feed (2 t
o 6 to 18)"
2580 DATA 134,"Apex up or down
(U/D)?"
2590 DATA 131,"Approx depth of print
(inches)?"
2600 DATA 134,"Obique/Vertical stripes
(O/V)?"
2610 DATA 131,"Unidirectional Mode
(Y/N)?"
2620 :
2630 DEF PROCfixBlock:PRINT TAB(35,0);
2640 @%=&0A
2650 lm$=FNip(4,-1,no$+"./")
2660 IF LEFT$(lm$,1)="." lm$="0"+lm$
2670 PRINT TAB(35,0)lm$:lm$=EVAL(lm$)
2680 IF lm <0 OR lm >7.9 VDU7:PROCfixBl
ock
2690 PRINT TAB(35,1);
2700 rm$=FNip(4,0,no$+"./"):rm$=EVAL(rm$
)
2710 IF rm <0 OR rm >8 VDU7:PROCfixBloc
k
2720 IF LEFT$(rm$,1)="." rm$="0"+rm$
2730 PRINT TAB(35,1)rm$:pbw=EVAL(rm$)-1
m
2740 rMargin%=INT((rm$pitch)+0.01)
2750 lMargin%=INT((lm$pitch)+0.01)
2760 chrsLne%=rMargin%-lMargin%
2770 VDU2,1,27,1,108,1,lMargin%
2780 VDU2,1,27,1,81,1,rMargin%,3:ENDPRO
C
2790 :
2800 DEF PROCshape
2810 PRINT TAB(38,2);:sh$=FNip(1,-1,"SR
")
2820 down=(sh$="R"):shap=(sh$="S")
2830 IF down PRINT TAB(35,9)s$;
2840 IF shap rect=FALSE
2850 ENDPROC

```

Patterns from a Keyboard

```
2860 :
2870 DEF PROCjustify:PRINT TAB(38,3);
2880 jstf$=FNip(1,-1,"LCR")
2890 left=(jstf$="L"):cent=(jstf$="C")
2900 right=(jstf$="R"):VDU2,1,27,1,97,1
2910 IF left VDU0,3
2920 IF cent VDU1,3 ELSE VDU2,3
2930 IF NOT text PRINT TAB(35,4)s$ TAB(
35,11)s$;
2940 IF cent PRINT TAB(35,11)s$;
2950 ENDPROC
2960 :
2970 DEF PROCjoinCrop
2980 IF shap PRINT TAB(35,4)s$;:ENDPROC
2990 PRINT TAB(38,4);:js$=FNip(1,-1,"JC
U")
3000 join=(js$="J"):crpd=(js$="C")
3010 unMo=(js$="U")
3020 IF crpd AND chrsLne% <LEN(A$) PROC
wrog:ENDPROC
3030 IF crpd PRINT TAB(35,11)s$;:ENDPROC
C
3040 IF NOT unMo PRINT TAB(35,5)s$ TAB(
35,6)s$ TAB(35,7)s$
3050 PRINT TAB(35,11)s$;:ENDPROC
3060 :
3070 DEFPROCwrog
3080 VDU7:PROCwnd(0,24,39,16,-1)
3090 PRINT TAB(0,0)y$ "Block too narrow
- Try again";
3100 PROCwnd(0,14,39,3,0)
3110 PROCnextStage:ENDPROC
3120 :
3130 DEF PROCgradBandUni
3140 PRINT TAB(38,5);:gbu$=FNip(1,-1,"B
U")
3150 unif=(gbu$="U"):band=(gbu$="B")
3160 IF unif PRINT TAB(35,6)s$ TAB(35,7
)s$
3170 IF band PRINT TAB(35,8)s$;
3180 IF shap AND band PRINT TAB(35,10)s
$;
3190 ENDPROC
3200 :
3210 DEF PROCincrDecr:PRINT TAB(37,6);
3220 bnd%=VAL(FNip(2,0,no$))
3230 PRINT TAB(38,7);:id$=FNip(1,-1,"ID
")
```

```
3240 incr=(id$="I"):decr=(id$="D"):ENDP
ROC
3250 :
3260 DEF PROCsetFeed
3270 IF unif PRINT TAB(37,8);:feed%=VAL
(FNip(2,-1,no$))
3280 IF shap PRINT TAB(35,10)s$;
3290 IF feed% <2 OR feed% >18 VDU7:PRO
CsetFeed:ENDPROC
3300 VDU2,1,27,1,65,1,feed%,3:ENDPROC
3310 :
3320 DEF PROCupDown
3330 PRINT TAB(38,9);:ud$=FNip(1,-1,"UD
")
3340 up=(ud$="U"):down=(ud$="D"):ENDPRO
C
3350 :
3360 DEF PROCoblqVert
3370 PRINT TAB(38,11);:ov$=FNip(1,-1,"O
V")
3380 vert=(ov$="V"):oblq=(ov$="O"):ENDP
ROC
3390 :
3400 DEF PROCdepth:PRINT TAB(35,10);
3410 depth=EVAL(FNip(4,-1,no$+"."))
3420 ENDPROC
3430 :
3440 DEF PROCtraverse
3450 PRINT TAB(38,12);
3460 IF FNip(1,-1,"YN")="N" prefix$="Bi
-":ENDPROC
3470 prefix$="Uni-":VDU2,1,27,1,85,1,1,
3
3480 ENDPROC
3490 :
3500 DEF PROCjoin:*FX3,10
3510 chrsBlk%=chrsLne%*ABS(FNfeeds(dept
h))
3520 numPhrs%=chrsBlk% DIV LEN(A$)
3530 rem%=chrsBlk% MOD LEN(A$)
3540 t%=0:REPEAT:t%=t%+1
3550 PRINT A$;:UNTIL t%>=numPhrs%
3560 PRINT LEFT$(A$,rem%):*FX3,0
3570 ENDPROC
3580 :
3590 DEF FNfeeds(dst)
3600 LOCAL vert%:s%=0:prev%=0:pFeed%=0
3610 IF unif d=dst*72 ELSE d=dst*216
```

```

3620 REPEAT:s%=s%+1
3630 IF band vert%=vert%+s%
3640 IF unif vert%=vert%+feed%
3650 IF vert% <d prev%=vert%:pFeed%=s%
3660 UNTIL vert% >=d
3670 IF num%=0 AND d-prev% <vert%-d end
%=pFeed% ELSE end%=s%
3680 =end%*-1
3690 :
3700 DEF PROCprintBlock
3710 count%=0:strip%=0:vert%=0
3720 maxLn%=FNfixLn:lineLen%=LEN(maxLn
e%)
3730 IF unMo line%=maxLn%
3740 IF band REPEAT:strip%=strip%+1
3750 IF band AND rect bndWid%=FNfeeds(d
epth/bnd%):spac%=bndWid%*decr
3760 IF shap spac%=lineLen%/bnd%*(-1*dec
r)
3770 IF up PROCupLoop
3780 IF down PROCdownLoop
3790 IF unif AND up ENDPROC
3800 IF band UNTIL strip%>=bnd%
3810 ENDPROC
3820 :
3830 DEF FNfixLn
3840 len%=LEN(A%)
3850 numPhrs%=chrsLn% DIV len%
3860 rem%=chrsLn% MOD len%
3870 =STRING$(numPhrs%,A%)+LEFT$(A%,rem
%)
3880 :
3890 DEF PROCupLoop
3900 IF unif strip%=1
3910 lim%=INT((lineLen%/bnd%)*strip%)
3920 REPEAT: PROCprintLine(count%,1)
3930 IF decr UNTIL spac%=0
3940 IF incr OR unif UNTIL LEN(line%)>=l
im%
3950 IF band AND incr spac%=0
3960 ENDPROC
3970 :
3980 DEF PROCdownLoop
3990 lim%=INT(lineLen%*((bnd%-strip%)/bn
d%))*-1
4000 IF unif AND crpd REPEAT
4010 IF band OR crpd OR unif chrsLn%=
chrsLn%+1

```

```

4020 REPEAT: PROCprintLine(chrsLn%,-1)
4030 IF rect AND band UNTIL spac%=bndWi
d%*incr
4040 IF shap UNTIL LEN(line%)<=lim%*band
4050 IF rect AND unif AND NOT crpd UNTI
L end
4060 IF rect AND unif AND crpd UNTIL en
d OR chrsLn%=rem%
4070 IF rect chrsLn%=lineLen%
4080 IF unif AND crpd UNTIL end
4090 ENDPROC
4100 :
4110 DEF PROCprintLine(chLn%,factor)
4120 chLn%=chLn%+factor
4130 IF incr spac%=spac%+1
4140 IF decr spac%=spac%-1
4150 IF spac% <0 spac%=0
4160 IF unif vert%=vert%+(feed%*3)
4170 IF vert% >=depth*216 end=TRUE
4180 IF band PROCshade
4190 IF oblq OR cent OR shap line%=FNsha
p
4200 IF rect line%=FNrect
4210 *FX3,10
4220 PRINT line%:*FX3,0
4230 VDU3:chrsLn%=chLn%:back%=vert%+un
if
4240 count%=chLn%:ENDPROC
4250 :
4260 DEF FNshap
4270 IF (right AND oblq) OR (vert AND l
eft) OR (cent AND shap):=LEFT$(maxLn%,c
hLn%)
4280 =RIGHT$(maxLn%,chLn%)
4290 :
4300 DEF FNrect
4310 IF crpd:=RIGHT$(maxLn%,chLn%)+MID
$(maxLn%,rem%+1,lineLen%-chLn%)
4320 =maxLn%
4330 :
4340 DEF PROCshade:vert%=vert%+spac%
4350 VDU2,1,27,1,51,1,spac%:ENDPROC
4360 :
4370 DEF PROCfinish:PROCwnd(0,24,39,17,
-1)
4380 VDU2,1,27,1,64,3
4390 PRINT y% "Key f4 to start again":E
ND

```

BEEBUG

Software for the BBC Micro and Master Series



Beebug C Programming Language

Beebug C is the much acclaimed C programming language for the BBC Micro and Master 128. Although normally only available on more powerful computers, Beebug C is a full implementation of the Kernighan & Ritchie standard. Features include:

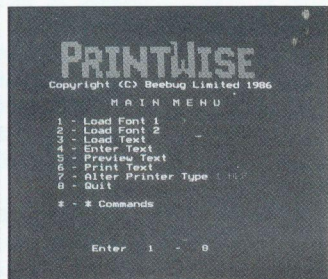
- ★ Runs on BBC B, B+ and M128.
- ★ Comprehensive set of ANSI functions.
- ★ OS functions vdu, osbyte, mode etc.
- ★ Command line interpreter.
- ★ Floating-point maths.
- ★ Linker for multi-source programs.
- ★ Expandable run-time library.
- ★ Optional stand alone generator.
- ★ Optional maths functions.
- ★ Full macro handling facilities.
- ★ Supplied on 2 ROMs & library disc

Normal price: **£60.28** inc VAT

Members price: **£45.21** inc VAT

Stock code: 0074 40T, 0075 80T

Please add £2.00 carriage.



Printwise

Printwise is a low-cost publishing aid allowing you to create professional looking magazines, leaflets, posters etc - the possibilities are endless. Simply take your text file, use embedded commands to specify the font styles you require, and let Printwise do the rest.

- ★ 9 authentic fonts from 4pt to 40pt.
- ★ Font designer for creating new fonts.
- ★ Fonts may be used on the same line.
- ★ Italics, bold, condensed, reversed etc.
- ★ Proportional spacing.
- ★ Subscript and superscript.
- ★ Left, centre, right & full justification.
- ★ Suitable for Epson compatible printers.

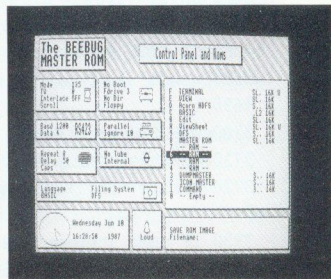
Printwise is easy to use and requires no programming skills. It may be used with text files created on Wordwise, View, InterWord, Mini Office and almost any text editor.

Normal price: **£30.66** inc VAT

Members price: **£22.99** inc VAT

Stock code: 0085 40T, 0086 80T

Please add £2.00 carriage.



Master ROM

The Master ROM is a powerful 32K ROM packed with features to enhance the facilities of the Master 128.

Disc Menu - A single command takes you to a full feature disc menu displaying all the items in the current directory. You can change directories or run, copy, delete, rename selected files.

Control panel - This displays all the computers status settings and the ROMs fitted. The cursor keys may be used to adjust any of the settings, which may be saved for future loading at any time.

Disc commands - A whole range of useful ADFS commands, including: *FIND, *FORMAT, *VERIFY, *BACKUP, *MERGE, *WIPE etc.

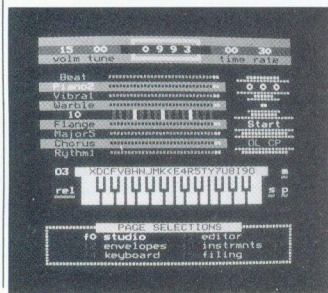
Plus: **16K-64K Printer buffer**
Simple 16K-64K RAM disc
Diary and automatic alarm

Normal price: **£39.84** inc VAT

Members price: **£29.88** inc VAT

Stock code: 0087 ROM

Please add £2.00 carriage.



Studio 8

Studio 8 is a real-time studio system with digital recorder, rhythm and drum machines which give hours of entertainment.

Studio - Allows playing and recording in real time with keyboard and sequencer.

Editor - A full-screen editor allowing the precise editing of notes.

Envelope editor - allows the definition of up to 16 amplitude and pitch envelopes.

Instrument definer - Create up to 32 instruments by combining pitch and amplitude envelopes, volume & sustain.

Plus many more features.

Normal price: **£22.48** inc VAT

Members price: **£16.86** inc VAT

Stock code: 0009 80T

Please add £2.00 carriage.

HINTS HINTS HINTS HINTS HINTS

and tips and tips and tips and tips and tips

This month we have a bumper crop of hints and tips, with particular thanks to David Holton. All hints and tips are most welcome, so do keep sending them in.

EXTRA HELP

David Holton

If you put a full stop after the *HELP command (i.e. *HELP.) then this is a signal to each of the ROMs to display their full help information. Some ROMs (like the DFS 2.24 ROM in the Master) output their full information once for each full stop, so entering *HELP ... will display extended help for all the ROMs and 3 sets of extended help for the DFS.

HIDDEN ROMS

David Holton

If, when writing a sideways ROM, you leave the title blank and give no binary version number, the ROM will not show up under *ROMS, although it will respond correctly in all other respects. Espionage agents, beware!

RANDOM ASSEMBLER

David Holton

A quick way to obtain random numbers in assembler is simply to read bytes from the ROM, especially from the address range &E000 to &FC00. It is a lot easier than writing a genuine random number generator, and is also more efficient.

ASSEMBLY ADDRESS

David Holton

If you open your assembly text with:

```
[  
OPT pass%
```

(i.e. with the [on a different line to the OPT statement) then the assembler displays the

value of P% at the start of each pass. If you wish to suppress the printing of P% (because you don't want a ragged start to a Basic program containing assembler, for example) then put the bracket and the OPT statement on the same line, thus:

```
[ OPT pass%
```

ASSEMBLER SQUARE BRACKETS

John Blackburn

If you enter a left hand (opening) square bracket at the Basic prompt followed by Return, then the current value of P% is printed in hexadecimal. This a useful shorthand for PRINT ~P% at the prompt. The reason that P% is printed is related to the "Assembly Address" hint above - the assembler starts up, prints the address it is to assemble at, but assembles nothing.

PRINTER CHECK

Stephen Wilcock

When a program requires the use of a printer, the following line at the beginning of a program will check that the printer (parallel or serial) is switched on, on line, and ready to receive data. This will prevent the computer hanging up at the crucial moment of printing.

```
10 VDU2:PRINT':VDU3:A=INKEY(10):IF ADVA  
L(-4)<63 PRINT "PRINTER IS NOT ON LINE"  
:END
```

LISTING VARIABLES

The following short function key definition will list the names of all the real variables used by a program. Note that only those variables actually used (and assigned to) when the program was run will be listed.

```
*KEY0 F.F%=&482TO&4F4S.2:A%=!F%A.&FFFF:  
IFA%>&FF REP.V.10,13,-512+F%/2:M%=A%+1:  
REP.M%=M%+1:V.?M%:UN.?M%=0:A%=!A%A.&FFF  
F:UN.A%<&FF:N.EL.N.|M
```

B

RISC USER

The Archimedes Magazine & Support Group

Now in its fifth year of publication, RISC User continues to enjoy the largest circulation of any magazine devoted solely to the Archimedes range of computers. It provides support for all Archimedes users at work (schools, colleges, universities, industry, government establishments) and home. Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines.

A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer to enthusiasts and professionals at all levels.

The Archimedes Magazine & Support Group

Here are some articles and series published in the most recent issues of RISC User:

ACORN LAUNCH A5000 SYSTEM

A review of the new breed of Archimedes, the A5000.

EXPLOITING THE WIMP

A new series looking at some practical aspects of Wimp programming.

RECOVERING LOST DATA

Recovering programs and data lost in the Desktop.

RENDER BENDER II

A look into the multi-tasking version of this popular ray-tracing package for Clares.

DRAW TOOLS

A toolbox for Draw allowing the precise creation of shapes such as circles, arcs, lines and sectors, at predetermined sizes and angles very simply.

ACORN'S MULTI-TASKING PC EMULATOR

A review of this powerful multi-tasking PC emulator.

SPASETECH'S FAX SCAN

A look at this inexpensive way of turning a fax machine into a scanner for the Archimedes.

FORM DESIGNER

An excellent application which makes the creation of printed forms and tables very easy indeed.

AN EXTENDED MULTI-FUNCTION DESKTOP CALCULATOR

A desktop calculator which offers a full decimal and hexadecimal keypad, calculations in binary, decimal and hex, Undo and Redo options, and facilities to save and reload displays.

USING ANSI C

A series of articles on programming Desktop applications in C.

WP/DTP

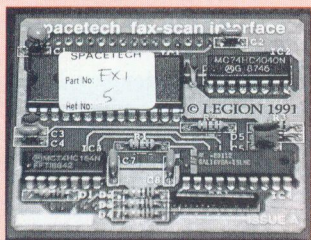
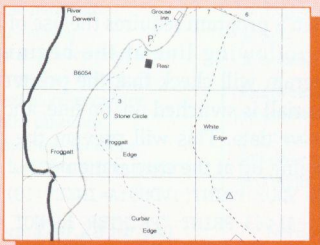
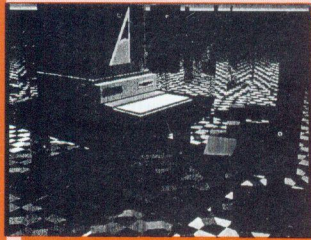
A regular column on using different DTP and WP packages. The latest article describes the steps needed to combine text and graphics using Edit and Draw to produce detailed maps.

ARCADE

A round-up of the latest games for the Archimedes.

INTO THE ARC

A regular series for beginners currently explaining how to use the Task Manager.



SUBSCRIPTION DETAILS

As a member of BEEBUG you may extend your subscription to include RISC User for only:

| | Additional Cost |
|-------------------------|-----------------|
| Destination | |
| UK, BFPO & Ch Is | £ 10.50 |
| Rest of Europe and Eire | £ 15.40 |
| Middle East | £ 19.60 |
| Americas and Africa | £ 21.90 |
| Elsewhere | £ 33.00 |



RANDOM NUMBER SAMPLING

I recently came across Mike Williams' article on random sampling (*Workshop*, BEEBUG Vol.10 No.4), which I read as I have a professional interest in this topic. There are a number of comments I would like to make.

It is slightly misleading to say that the normal density function cannot be integrated - what is meant is that it does not have an integral which can be expressed as a simple combination of elementary functions (roots, algebraic, trig, exponential and logarithmic functions). In fact, this integral (with a little juggling of the limits and constants) is the definition of the error function, and of course, it can always be integrated numerically.

It would be nice to see some estimate of the frequency with which the rejection sampling generator does reject numbers. This could be easily estimated using a test harness which totals up the number of times line 1030 is executed when, say, 100 normal random numbers are generated.

The article omits one of the more interesting standard normal random number generators, the *Box-Muller* method. Sample code for this generator is:

```
DEF FNbm_normal
LOCAL x1, x2
x1=SQR (-2.0*LOG (RND (1)))
x2=SIN (2.0*PI*RND (1))
=x1*x2
```

This generator is derived by observing that though the normal density function does not have an integral which can be expressed simply, the product of two normal density functions does have a simple integral when the variables are changed from Cartesian to polar co-ordinates.

Ron Larham

My thanks to Mr.Larham for his helpful comments. With reference to Mr.Larham's first

point I must admit my terminology was a little loose - I did in fact mean that it could not be integrated analytically.

The Box-Muller method described above is one I considered for inclusion in the article, but judged that the mathematical explanation might be too much like hard going for some readers, and so decided to omit it. Readers might like to incorporate the function given by Mr.Larham into the program listed in the article to compare its effects with the other normal random number generators.

GAMES AND APPLICATIONS

I would like to add to the sentiments of Rowland Fraser (*Postbag*, BEEBUG Vol.10 No.4). I have only recently bought a secondhand BBC. I bought as many of the BEEBUG back issues as I could and have found them to be excellent. Among the programs I have typed in have been Jigsaw (Vol.4 No.3), Fancy Lettering (Vol.5 No.10), and Route Planner (Vol.7 No.6). I enjoy the puzzle types of games you have published recently, and I find games do stimulate an interest with my children. I think generally the balance is very good.

Incidentally, I hope that you will cover in the *First Course* series, examples like the following:
 $X\% = X\% - 10 * (J\% = 13) * (X\% > 10) + 10 * (J\% = 14) * (X\% < 1268)$

I believe that it is to do with TRUE and FALSE, but I need more help.

I also enjoyed the *Workshop* in Vol.10 No.4. The challenge of seeing old formulae I still can't grasp was most stimulating.

Elaine Kemp

In fact, and this is no more than coincidence, this month's First Course deals with the use of logic in expressions like the one above. The fact that TRUE is represented by -1 and FALSE as 0 can be very useful as the article shows. **B**

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 40p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 5th of each month.

Assorted magazine discs £1.50 each or 18 for £23, please phone for details. BEEB Video Digitiser from Watford Electronics hardly used £40, Viglen keyboard and computer console for BBC £15, Arabic-English system ALNOOR for BBC micro, plus complete set of Arabic alphabet keyboard covers £30, Printer stand legs £3, Eprom Programmer MK11 for BBC W.E. £20, Printmaster ROM with instruction book £6, AMX Pagemaker ROMs plus users guide hardly used £20, disc storage box for 5.25" discs, with lock, holds 50 discs £3.50. Tel. 081-346 7228, wk/days after 4pm or all day wk/ends.

BBC B issue 7 with DFS, twin 40/80T disc drives, Black & White monitor, Sideways ROM fitted into a Viglen Console, AMX Super Mouse, Cumana Sketch Pad, View word processor, many tapes, discs & books £300 o.n.o. Tel. (0372) 375438 eves.

Acorn Cambridge Workstation, 6502 and 32 bit processors, 4Mb RAM, 12" high res. colour monitor, 5.25" floppy drive, hard drive, (faulty), in one unit with detachable keyboard, also usable as model B, includes 32000 Assembler, BBC Basic, C, Cambridge Lisp, Fortran 77, ISO Pascal, offers (or swap for Arch.) Tel. 081-698 3772.

BBC Master 128 with 512 co-processor upgraded to 1Mb, DO6 2.1 twin 40/80 Cumana disc drives, Microvitec RGB colour monitor, twin joysticks, Modem, 30 plus computer books, 100's computer magazines, software, EPROM software, EPROM Programmer and UV Eraser, Computer desk, 3' 4" wide 2' deep 3' high on castors, plus many extras £500. Tel. (0243) 265392.

BBC Micro issue 3 with 'Replay' ROM and DFS £120, View ROM £10, monitor

stand £5, BEEBUG magazine volume 9 £4. Tel. 90734) 791436.

Early BBC WD1770 ADFS (perfect) £100, Watford ROM/RAM card 96k SRAM 16k battery backed £60, co-pro 512k with mouse DOS+ and GEM (original) £180, Essential Soft upgrade 896k £90, Dual 40/80T discs in Bridge mounting £80, 12" Amber monitor £30, Wordwise + and Word-ex £25, Inter-Word £20, Genie Watch £15, Joystick £10, BEEBUG 'C' £20, loads of discs, spares, handbooks, see working £500 the lot, for more info Tel. (0253) 823462.

M128 with Turbo co-processor, dual Opus disc drives, all manuals and support discs, reference manuals part 1 & 2, Wordwise Plus and Spell Master ROMs all in original packaging £350 or will consider splitting. Tel. (0227) 262750.

Care Master Smart Cartridge; allows interruption of any program to execute various built in or user defined functions. The cartridge has hardly been used and is complete with original instructions £20. Tel. 081-654 3733.

Issue 7 BBC B selling separately, with Acorn DFS/DNFS with ROM board £100, dual 40/80T drives £80, Microvitec 1431 colour monitor £80, Acorn 6502 second processor with Hi Basic & manual £50, Dabs Hyperdriver, Mini Office II, Dabs guide to Mini Office II, Fleet Street Editor, Edword, Wordwise Plus, Wordaid, Slave+, User Dump, Keyword, 3 games, twin joysticks, several books on assembly language etc, disc storage box with some discs, solidisk DDFS no manual, please telephone for list & asking prices. Tel. 061-427 2144.

BEEBUG magazine tapes ideal for copying onto disc, Vol. 2 No. 9, Vol. 3

No. 1,2,4,5,6,7,8,9,10 Vol. 4 No. 3,4,5, Vol. 5 No. 6,9,10, Vol. 6 No. 1,4,7,10, 50p each, Knitwear Designer and Mini Office II 80T disc £5 each, Slave EPROM £10, Kourtyard, Crystal Castles, Krackout, Icarus, Village of Lost Souls all 80T discs £4 each, postage extra. Tel. 061-789 0500 after 6pm.

WANTED: BBC B software on 40T discs, my collection of BEEB software has been destroyed, would particularly like games, any age, Planetoid, Rocket Raider, Play it Again Sam's etc, in original packs, Please send list of what you have and how much wanted to; Mr R Jones, PO BOX 14455, Jeddah 21424, Saudi Arabia or fax; 010 966 2 648 3314, will reimburse postage/fax expenses when purchase.

AMX Mouse plus Stop Press DTP package complete with all manuals, discs and ROMs for BBC B & B+ in ex. cond. £25, also Interbase, Database for BBC B & B+ complete with ROM, manual and disc £25, please allow also £1 for P&P. Tel. (049525) 4889.

Interbase £20, BEEBUG Mag Vol. 1 to present complete £40. Tel. 081-291 2633.

64k Turbo Electron fitted with Jafa shadow RAM board, Slogger ROMbox Plus with printer interface, P.R.E.S ADFS plug in interface, View & Viewsheets cartridges, all manuals and some other books £160 the lot. (0483) 63470.

M128 with 6502 (turbo) co-processor, dual 80T disc drive, 5.25" in plinth, Citizen LSP10 printer, Viewspell, all manuals, BEEBUG Mags Vol. 2 complete, some Vol. 1 other programme books, joysticks, 15 programmed discs, 8 blanks in holder, 5 tapes, black ash effect desk/station holds all (23" X 29" X 28") £500 the lot

o.n.o. Tel. (0533) 886904 eves/wk/ends (buyer collects).

BBC B issue 7 Acorn DFS, 80T Watford disc drive with power supply, Acorn 6502 second processor with DNFS and hi-basic ROMs and manual, Microvitec CUB 895 colour monitor, Acorn cassette data recorder, pair BBC joysticks, pair Voltmace joysticks with splitter box and driver program, Watford Speech synthesiser with ROM and manual, Polarizing filter for Microvitec monitor, over 35 software titles all in original boxes with manuals including software on ROM, disc and cassette, also 10 books, over 40 issues of BEEBUG, 20 issues of RISC User and more than 30 issues of Acorn User plus other assorted magazines. Phone for a list or make me a reasonable offer. Tel. (0277) 654343 between 5pm and 10pm.

M128, 2x 5.25" DSDD, Modem, Mouse & Holder, 30x 8K/16K EPROMS, Morley Master board AA, Dumpout 3, Spellmaster, Interword, Disc Boxes, Welcome Guide & disc, Ref books include; Disc Companion, Understanding I/Word, Master Operating System, Century Computer prog. Course, Complete User Handbook, Graphics on BBC, Advanced User Guide, Advanced Disc User Guide, Discs include; Fax-File Organiser, Fancy Label Suite, Keyword, Magscan, Morley ATS Utilities, Crossword Compiler (Panda), Family History, Scapeghost, AMX Super Art, File Handling (BEEBUG) Disc User discs, and some AU discs, all with documentation/manuals £550 o.n.o. Tel. (0442) 826079 after 6pm.

BBC B with DFS, dual 80T drives, CUB colour monitor, desk, software including View Professional £260, will split. BBC B issue 3 including Acorn 1770 DFS for spares £75, Panasonic KXP-1124 printer £160. Tel. (04867) 80632.

Acorn 21Mb hard disc & utilities disc £200, 512 DOS Plus processor, 1Mb essential software upgrade, Acorn Mouse, 11Mhz XTAL, all original discs, manuals and software, mint condition £250, BBC Master reference manuals Nos. 1 & 2 £10, Master 512 User & Technical Guides & discs £20, latest Master O.S. ROM £15, Problem Solver,

Tulls mouse driver, Compactor & Essential 512 Util discs £5 each. Tel. (0326) 240734.

BBC B & DNFS MAX ROM, green & colour monitors, 40T disc drive, switchable disc drive, mouse, AMX Super Art, EPROM Programmer & Eraser £350 o.n.o. Tel. (0323) 410737 wk/ends only.

WANTED: Correspondence from owners of BBC or Master computers who wish to exchange ideas, especially on games software. Write to: Mr Joseph Gatt, Dwejret iz-zahar, Triq, Sigmundou, Dimech, Balzon, MALTA.

BBC B ADFS twin 80T drives, CUB colour monitor Wordwise+, View Professional BBC Graphics ROM Epson printer Morley 20Mb SCSI hard disc drive, computer desk £600 will split. Tel. (04867) 80632.

FABLE **Fast Advanced Basic Line Editor**

featuring...

Word Processor editing of Basic programs
Sophisticated menu system
Simple control-key alternatives
Easy customisation and linkage with basic Interword and View editing modes
Desktop Save/Load
Search, Copy, Delete, Renumber, Print etc.
ONLY £4.99 for unprotected sideways RAM image on disc, source code and brief instructions.

Written by Richard Taylor, author of BEEBUG magazine's Monix.

**1 Ulverston Road, Dunstable, Beds, LU6 3QE.
Tel. (0582) 606021 (eves) for more information.**

M128, with 512 co-processor, DOS+ 2.1 with GEM software, Acorn 14" colour monitor, swivel/tilt monitor base, monitor bridge, Cumana 40/80T DS 5.25" drive with PSU, Epson LQ800 24 pin printer, Vine Micros ROMboard 3, Master Replay with OS 1.2, EPROM Programmer & Eraser, 7 additional ROMs including Questpaint & mouse, 46 games, 100 capacity disc box, 80 discs, 5 additional manuals and books, AU's from November '86 to present, BEEBUG magazines from May '87 to present, all in original boxes, complete system, all good condition, upgrade forces sale £750 o.n.o. phone for complete list. Tel. (0900) 68231.

M128 with 80186 co-processor including View, Word, Gem, Editor and Terminal Emulator, Mouse and all

manuals, leads etc. £400, Cumana CS400S and CS400 5.25" disc drives, £135, Zenith Data Systems 14" green monochrome VDU £50, software Mini Office II, Printer Driver, Graphics, NLQ and Dumpmaster on chip or disc, spare discs, BEEBUG magazines, books on basic programming £50, prefer to sell the lot for £600. Tel. (0632) 62782.

WANTED: Compact companion interface unit for Master compact, this unit was produced by the MERTEC company, I would like all instructions and software if possible, if you have one sitting around doing nothing give me a call. Tel. (0323) 846068 anytime.

Viewstore & Viewspell both in original packaging with manuals £15 each or £25 for both, plus postage, little used replaced by Overview, Electron with Plus 1 and AP3 3.5 disc drive and ADFS interface some educational programs, offers to David. Tel. (0792) 865691.

Modem - (Pace 'Nightingale') with BEEBUG 'Command' communications ROM £45 inc. postage. Tel. (0294) 52250.

Z88 Cambridge portable computer, 128k RAM pack, mains adaptor, carry case, user guide only £70. Tel. (0679) 62728.

Test Gear, Wayne Kerr Universal Bridge B221 with low impedance adaptor £60, Solartron Precision ac Millivoltmeter

VF252 1.5mV - 15V £50, 3 scrap Avometers for spares free to collector. Tel. (0483) 573688.

Master 128, Microvitec CUB colour monitor, Pace 5.25" DD 40/80 disc drive, all View family, all manuals, Master ROM, Hyperdriver, joysticks etc. £350 or v.n.o. Tel. (0793) 870682.

BBC B with Watford DDFS, dual 3" drives & Acorn SS 5.25", Acorn Prestel & Teletext, View, Viewsheet, Viewstore, Speech chip, all manuals, many blank discs, games & serious software, "Facit" 9 pin dot matrix printer (inc. View driver) £300 the lot. Tel. 071-249 1482.

Watford dual disc drive 5.25" 40/80T & PSU £70. Tel. 081-318 5155.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

| | |
|--------|-----------------------------------|
| £18.40 | 1 year (10 issues) UK, BFPO, Ch.1 |
| £27.50 | Rest of Europe & Eire |
| £33.50 | Middle East |
| £36.50 | Americas & Africa |
| £39.50 | Elsewhere |

BEEBUG & RISC USER

| |
|--------|
| £28.90 |
| £42.90 |
| £53.10 |
| £58.40 |
| £62.50 |

BACK ISSUE PRICES (per Issue) 1 July 1991

| Volume | Magazine | 5"Disc | 3.5"Disc |
|---------|----------|--------|----------|
| 6 | £1.00 | £3.00 | £3.00 |
| 7 | £1.10 | £3.50 | £3.50 |
| 8 | £1.30 | £4.00 | £4.00 |
| 9 | £1.60 | £4.75 | £4.75 |
| 10 | £1.90 | £4.75 | £4.75 |
| Binders | £4.20 | | |

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

POST AND PACKING

Please add the cost of p&p when ordering individual items. See table opposite.

| Destination | First Item | Second Item |
|-----------------|------------|-------------|
| UK, BFPO + Ch.1 | £ 1.00 | £ 0.50 |
| Europe + Eire | £ 1.60 | £ 0.80 |
| Elsewhere | £ 2.40 | £ 1.20 |

BEEBUG
 117 Hatfield Road, St.Albans, Herts AL1 4JS
 Tel. St.Albans (0727) 40303, FAX: (0727) 860263
 Manned Mon-Fri 9am-5pm
 (24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by RISC Developments Ltd.

Editor: Mike Williams
 Assistant Editor: Kristina Lucas
 Technical Assistant: Mark Moxon
 Production Assistant: Shella Stoneman
 Advertising: Sarah Shrive
 Managing Editor: Sheridan Williams

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, RISC Developments Limited.

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud. In all communication, please quote your membership number.

RISC Developments Ltd (c) 1991

Printed by Arlon Printers (0923) 288328 ISSN - 0263 - 7561

Magazine Disc

November 1991
DISC CONTENTS

PATTERNS FROM A KEYBOARD - this application allows you to produce and print a huge variety of text patterns, created from the keyboard.

RUNNING PROGRAMS WITH A JOYSTICK - two routines which you can incorporate into a Basic program so that you can use it with a joystick.

BEEBUG WORKSHOP: SIMULATION MODELLING (2) - three programs demonstrating the event centered approach to simulation modelling, as well as two variations discussed in the article.

PRINTER SPOOLER UTILITY - this handy utility for printing text files in the background was previously published in Vol.3 Nos. 3 and 7, and has been updated to work on the Master series as well.

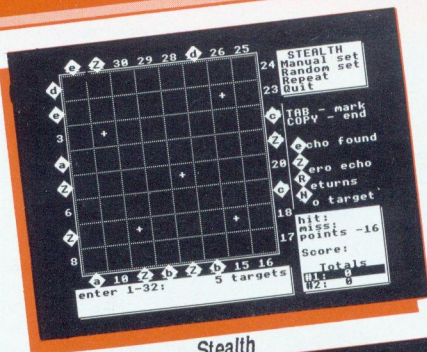
TIMED INTERRUPT ROUTINE - this utility allows you to interrupt any program in order to initiate a screen dump or any other machine code task.

STEALTH - in this challenging puzzle-game you are the Array Laser Location Operator and your task is to destroy the 'Stealth' attack force, which is invisible to radar.

WORDWISE USER'S NOTEBOOK: WORDWISE PLUS TELETEXT EFFECTS (2) - some more segment programs for users of Wordwise Plus illustrating the use of Teletext mode.

TEXBASE : A FREE FORMAT INFORMATION SYSTEM (PART 3) - a complete suite of all the programs for Texbase, including this month's *Setup* listing, which provides a complete free format information system.

MAGSCAN DATA - bibliography for this issue



Stealth

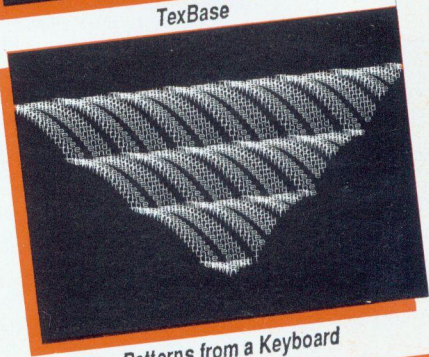
ENTER THE PAGE TO BE DELETED 1
TexBase Part 3

The process works in mode 7, to make more use of memory. It is necessary to know the page number to be deleted, as there is no keyword search facility.

The first few lines of the page are displayed for confirmation. However, because mode 7 uses a 40 character screen, some words will be split - this is not important.

IS THIS THE PAGE?

TexBase



Patterns from a Keyboard

ALL THIS FOR £4.75 (5.25" & 3.5" DISC) + £1 P&P (50P FOR EACH ADDITIONAL ITEM)
Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1) available at the same prices.

DISC (5.25" or 3.5") SUBSCRIPTION RATES
6 months (5 issues)
12 months (10 issues)

UK ONLY
£25.50
£50.00

OVERSEAS
£30.00
£56.00

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

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

The New Acorn A5000



A5000 Features

- RISC OS Version 3
- ARM 3 For Unbelievable Speed
- 1.6 Mb New Format Floppy Drive
- 40 Mb IDE Hard Drive
- Acorn Multi-Scan Monitor

The New Learning Curve Pack

- New Multi-tasking PC Emulator
- Genesis 2 Database
- 1st Word Plus Wordprocessor
- Acorn DTP
- Lemmings and Pacmania Games
- Audio Training Tape
- Optional 300 dpi Ink Jet Printer

Acorn have just announced the release of the new A5000 computer. It is the latest in the Archimedes range and features the new version of RISC OS (not currently available on any other machine).

Based around the ARM 3 processor running at 25 MHz, this machine goes like lightning.

With a price of £1499 (plus VAT) for the A5000 with ARM 3, Multi-Scan Monitor, 2 Mb RAM and a 40 Mb IDE hard drive this machine really is excellent value.

A brochure on the A5000 is available free on request, please phone or write.

If you have been considering upgrading your system, there has never been a better time. However, the A5000 is in very short supply and so if you wish to be one of the first in the country with this new system, please use the form below to reserve yours now. Orders will be processed strictly in order of receipt. Should you need to change or cancel your order at any time until the system is despatched this will of course be no problem. Cheques will not be cashed and cards not debited until your system is sent.

Beebug are one of Acorn's largest dealers and have significant quantities of the A5000 on order. You may rest assured of our commitment to the Acorn marketplace and to our customers. Our full technical support team will be available to assist you with any problems that you may have in coming to grips with this new computer.

Voted 'Dealer Of The Year 1990' by A & B Computing (now Archimedes World) readers for customer service.

BEEBUG A5000 Computer Reservation

Please Reserve For Me (delete as appropriate)

Price INC VAT

- | | |
|--|----------|
| A) One 2 Mb A5000 with 40 Mb Drive & Multi-Scan Monitor | £1761.33 |
| B) One 2 Mb A5000 Learning Curve With 40 Mb Drive & Multi-Scan Monitor | £1799.00 |
| C) One system B (as above) with Acorn Ink Jet 300 dpi Printer | £2075.00 |

I wish to trade in the following equipment, please send me a quotation _____

Cheque enclosed, value £ _____ Debit My Access / Visa / Switch Card (No. as below)

Card No _____ / _____ / _____ / _____ Expiry Date _____ / _____

Please Deliver The System (NOTE: Add £9.00 courier charge) _____ I Will Collect from showroom _____

NAME _____ MEMBERSHIP NO _____ SIGNED _____

DAY TEL NO _____ ADDRESS (if Different) _____

SEND TO: A5000 Reservation, BEEBUG Ltd, 117 Hatfield Rd, St. Albans, Herts AL1 4JS