

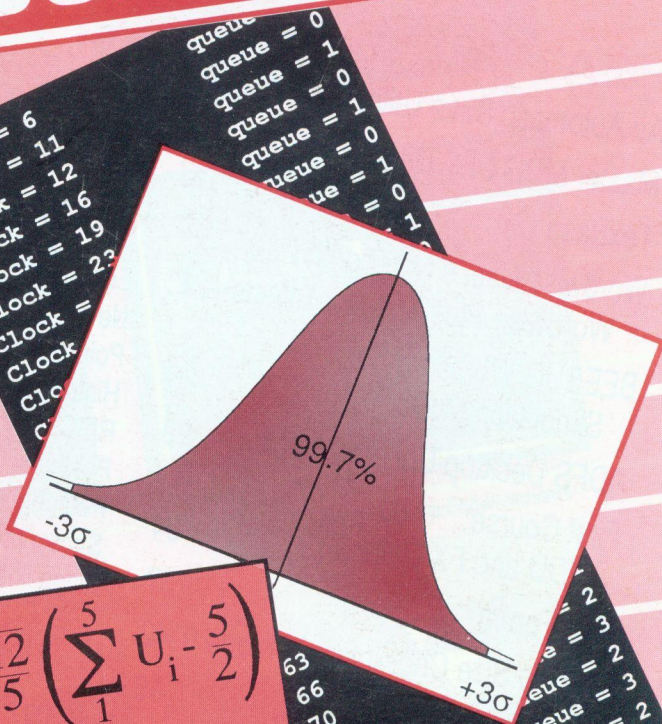
Vol.10 No.5 October 1991

BEEBUG

FOR THE
BBC MICRO &
MASTER SERIES

Simulation
Modelling

$$X = \sqrt{\frac{12}{5}} \left(\sum_{i=1}^5 U_i - \frac{5}{2} \right)$$
$$F(x) = \frac{1}{\sqrt{2P}} \int_{-\infty}^{+\infty} e^{-\frac{1}{2}x^2} dx$$



● AUTOMATIC SAVE UTILITY ● BBC PD SOFTWARE

● NUMERICAL NOTATION ● ADFS DESKTOP ENHANCEMENTS

FEATURES

An Automatic Save Utility	9
Master MOS DFS Bugs	14
TexBase (2)	16
Wordwise User's Notebook:	
Wordwise Plus Teletext Effects	21
BEEBUG Workshop:	
Simulation Modelling	25
ADFS Desktop Enhancements	29
First Course:	
VDU and FX Calls (6)	34
512 Forum	37
Double Size Characters in Mode 0	42
Numerical Notation	46
BEEBUG Function/ Procedure Library (6)	48
Hexagons	50

REVIEWS

BBC PD Public Domain Library	7
BEEBUG Education: Selladore Tales	44

REGULAR ITEMS

Editor's Jottings	4
News	5
Points Arising	55
Hints and Tips	57
RISC User	58
Postbag	59
Personal Ads	60
Subscriptions & Back Issues	62
Magazine Disc	63

HINTS & TIPS

Using Function Key 10	
Floating Point Rounding	
Formatting Help in Edit	
Function Keys in View	

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

Basic Auto Save Utility

Program name : CatDir
 Start line number : 10
 Incremented by : 10
 Save interval (mins) : 10
 Screen mode : 7

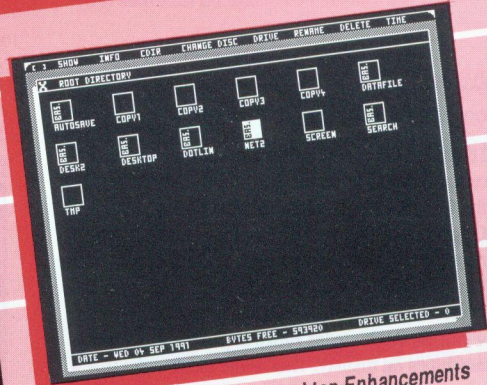
```

    ) Page:
    From:
    TITLE :TexBase
    line:=====
    1 Last month I introduced you to the idea of TexBase as a system to store and
    2 retrieve text based data. If you typed in the listings given, you should
    3 have been able to enter and store text. If you did enter some text, you
    4 would have found that you could not actually do anything with it at
    5 the time. However, we will solve that problem now by moving on to the
    6 important part, the search options.
    7
    8
    9
    10
    11
    12
    13
    14
    15
    16
    17
    18
    19
    20
    (R)ext (E)xit (P)rint (S)ave
    
```

TexBase

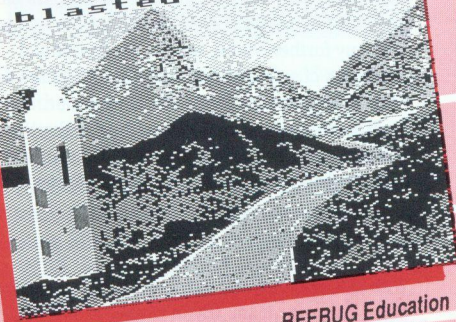
Automatic Save Utility

BASIC	Standard	Engineering
7E-5	7x10 ⁻⁵	70x10 ⁻⁶
7E-4	7x10 ⁻⁴	700x10 ⁻⁶
7E-3	7x10 ⁻³	7x10 ⁻³
7E-2	7x10 ⁻²	70x10 ⁻³
0.7	7x10 ⁻¹	700x10 ⁻³
7	7	7
70	7x10 ¹	70
700	7x10 ²	700
7000	7x10 ³	7x10 ³
70000	7x10 ⁴	70x10 ³
700000	7x10 ⁵	700x10 ³
7000000	7x10 ⁶	7x10 ⁶
70000000	7x10 ⁷	70x10 ⁶
700000000	7x10 ⁸	700x10 ⁶
7E9	7x10 ⁹	7x10 ⁹
7E10	7x10 ¹⁰	70x10 ⁹

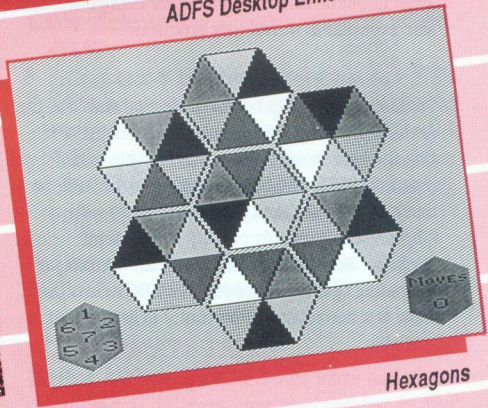


ADFS Desktop Enhancements

The rocks are
blasted clear!!



BEEBUG Education



Hexagons

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.

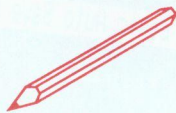


Program needs at least one bank of sideways RAM.



Program is for Master 128 and Compact only.

Editor's Jottings



BBC ACORN USER SHOW

The major show of the year for all owners of Acorn computers has to be the *BBC Acorn User Show* at the Wembley Conference Centre, London, 11th-13th October. And this year's show looks like being a major event. Although we have no firm details at the time of going to press, there is every indication that Acorn will be making some major announcements about the Archimedes range at the show, with new machines on display. That alone must make this a show not to be missed. Many other companies are also expected to unveil new products.

These days, it has to be acknowledged that it is the Archimedes range which hogs the limelight, and this is where most of the action is likely to be. However, BBCs and Masters will no doubt still be much in evidence, and discerning users with diligence can probably pick up some useful bargains. At the same time, if you are thinking of upgrading to an Archimedes, the *BBC Acorn User show* is a marvellous opportunity to see all that is available.

We shall have a total of three adjacent stands, one devoted to Beebug Retail, one devoted to our own hardware and software developments, and one for BEEBUG and RISC User magazines and associated products. These are stands 31, 32 and 33. All BEEBUG readers are most welcome to visit our stands, chat with staff, and see the latest product developments.

Remember that all our own products are offered at a discount to members, just one more reason for renewing your subscription if you haven't already done so. Our collections of programs (applications, utilities, games, etc.) and products like ASTAAD, MagScan and Filer offer real value for money.

RISC DEVELOPMENTS

As part of a purely internal move to foster future expansion, all magazine, book, software and hardware developments have, since 1st April, been incorporated in a new company called *RISC Developments Ltd.* Beebug Ltd., as a company, will henceforth deal only with retail sales. In future, therefore, you will see the RISC Developments name as publisher of BEEBUG and RISC User magazines (and any further books), and as developers and producers of software and hardware. There are no adverse implications for BEEBUG magazine, and we shall continue publication for as long as there is a demand for the magazine, now the only magazine exclusively devoted to the BBC micro, Master and Compact.

With this issue we also say farewell to two members of the magazine team, Glynn Clements our highly knowledgeable Technical Assistant, and hard working Technical Editor Alan Wrigley. However, we expect that Alan will continue to contribute regularly to BEEBUG. Mark Moxon has taken over as our new Technical Assistant.
M.W.

ACORN LAUNCHES NEW ARCHIMEDES

Acorn has launched a major addition to the range of Archimedes computers in the form of the A5000. Two models have been announced, though only one, a 2 MByte RAM, 40 MByte hard disc system, will be available immediately. The new system uses an ARM3 processor for performance substantially faster than the existing A3000 and 400 series, though the A540 remains the flagship of the Archimedes range.



The A5000 comes in a completely redesigned box, and the system will be sold complete with a multiscan high resolution colour monitor for £1499 (ex. VAT). The A5000 also features RISC OS 3, the latest version of the Archimedes operating system, with substantial improvements over the existing RISC OS 2. All main applications are now in ROM, outline font and printer support have been significantly rewritten, and there are many other enhancements which collectively increase the flexibility and control available to the user.

Acorn has also announced an A5000 Learning Curve package comprising an A5000 with 1st Word Plus word processor, Genesis Plus information system, Acorn DTP and the latest PC emulator with DR DOS, the whole package costing £1531 (ex VAT).

The new system will be demonstrated at the BBC Acorn User Show, and will be available for purchase from early October. For more information contact your Acorn dealer (this includes Beebug) or Acorn Computers Ltd., Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN, tel (0223) 245200.

MORE FOR 512 USERS

Essential Software, supporters of the 512 co-processor option for BBC micros, has announced six new products. All six should be available by the end of October:

1. Hard disc partitioner £10.95
2. File-find utility which will search up to four drives (including winchester) for a specified filename (including wild card characters) £12.95.
3. File-unerase, allowing recovery of a deleted file £11.95
4. Miscellaneous Disc II with a variety of utilities £14.95
5. Printer buffer program allowing use of 512's RAM in native mode, supplied in EPROM at £14.95
6. Enhanced version of CPFS which includes a printer buffer. Supplied in EPROM for £29.95, upgrades to existing users £7.00.

For more details contact Essential Software, P.O.Box 5, Groby, Leicester LE6 0GQ.

ALL FORMATS FAIRS

Further All Formats Computer Fairs have been organised as follows (see also the list in last month's News):

- 3rd Nov Royal Horticultural Hall, Westminster
- 10th Nov National Motorcycle Museum, Solihull
- 1st Dec City Hall, Candleriggs, Glasgow
- 14th Dec University of Leeds Sports Centre, Calverley Street, Leeds
- 15th Dec The Brunel Centre, Bristol Old Station (next to Temple Meads)



* THE SHOW *

*A unique Show dedicated to the world of Acorn and BBC computers with both existing and potential users in mind.

*A Show which brings together the best and latest in the Acorn market ranging from the widely used and much loved 8-bit computers right through to the up-to-the-minute 32-bit machines.

*A Show fully supported by Acorn Computers themselves who will be sponsoring many of the technical and informative activities.

*A Show presented in association with BBC Acorn User Magazine, a long established and respected publication, well informed on BBC and Acorn computer matters.

*A Show designed to entertain as well as provide opportunities for purchasing a wide variety of hardware and software, often at special Show prices.

* A Show NOT to be missed!! *

ACORN USER SHOW '91



* THE FEATURES *

*Fantastic Theatre providing on-going product information, computing techniques and tips, entertainment, competition prizes, celebrity spots and much more, all with a light hearted flavour.

*Personal appearance by CAROL VORDERMAN all day Saturday and LINFORD CHRISTIE and friends on other days subject to sporting activities.

*Computer problem "Clinics".

*Fabulous Games Arcade and High Score competitions. (Try the VERY LATEST games for the Archimedes).

*BBC Acorn User Magazine "Editor's Office".

*Schools Projects including in-house production of the unique BBC Acorn Show newsletter.

*Club Corner.

*Mock Classroom with Schools' activities.

*Rest-Your-Legs lounge area.

*Competitions.

*FREE Show Guide with all adult tickets.

MUCH, MUCH MORE

* THE VENUE *

There are fewer places in the nation better known than Wembley. Renowned for its Stadium, Event, Exhibition and Conference facilities, the BBC Acorn User Show '91 will provide both visitors and Exhibitors alike with a first class venue.

Excellent Rail, Road and Air links make it easy for visitors to reach Wembley. Massive on-site parking (NCP) is available for those travelling by car and a free shuttle bus will operate from Wembley Park Underground Station.

A variety of snack bars and restaurants will provide light refreshments throughout the open days.

* THE EXHIBITORS *

*Exhibitors represent many of the leading companies in the Acorn world. You will find;

*Software houses and distributors with programs covering the range from games right through to sophisticated desk-top publishing or image processing.

*Acorn appointed dealers with their own special offers and deals, and some from Acorn Computers themselves.

*Special peripheral manufacturers and distributors.

*Publishers and Clubs.

*Consumables and Accessories.

MANY MORE

* TIMES AND PRICES *

Tickets at Door;

Adult	£6.00
Under 16	£4.00
Family (2 Adults & 2 Children)	£16.00
Under 5's	FREE

Advance Tickets - use coupon below

FRIDAY 11th October 1991	- 10.00 a.m. - 6.00 p.m.
SATURDAY 12th October 1991	- 10.00 a.m. - 6.00 p.m.
SUNDAY 13th October 1991	- 10.00 a.m. - 5.00 p.m.

Please note - Pushchairs are not allowed in the Exhibition Halls.

* THE CONCERT *

A FABULOUS 60's CONCERT will take place in the Grand Hall of the Conference Centre on Friday evening (11th October 1991) with legendary groups HERMAN'S HERMITS, and THE BRUVVERS with the great MIKE BERRY.

Bring back the memories, sing along with your old favourites and generally let your hair down. See the Exhibition during the day and stay on for the Concert in the evening. DON'T MISS THE FUN!

Why not see the Concert and stay in London overnight and visit the Exhibition on Saturday. A great way to enjoy a trip to the Big City.

Concert tickets are all one price at a modest £10.50 each and are available from the Wembley Box Office - Tel. 081 900 1234.

ADVANCE TICKET APPLICATION FOR THE BBC ACORN USER SHOW

Please Rush me; Adult Tickets at £5.00 Advance Discount Rate
..... Under 16 Tickets at £3.00 Advance Discount Rate
..... Family Tickets at £13.00 Advance Discount Rate

I enclose Cheque/Postal Order for £..... made out to SAFESELL EXHIBITIONS LTD

NAME

ADDRESS

..... POSTCODE

RETURN THIS FORM TO SAFESELL EXHIBITIONS LTD, MARKET HOUSE,
CROSS ROAD, TADWORTH, SURREY, KT20 5SR TEL. 0737 814084


WEMBLEY
CONFERENCE AND
EXHIBITION CENTRE

BBC PD Public Domain Software Library

Peter Rochford discusses a new venture in public domain software for the BBC micro.

Product	Public Domain Software Library
Supplier	BBC PD Alan Blundell, 18 Carlton Close, Blackrod, Bolton BL6 5DL.
Price	£1.50 per 5.25" disc (inc. p&p) £1.75 per 3.5" disc (inc. p&p)

Perhaps it would be prudent first of all to explain for those who have not come across the term what 'Public Domain software' means and what it is all about.

Public Domain software is any software that is generally free of usual copyright constraints. This in effect means that an author puts no restrictions on the use of (or copying of) his/her software by any member of the public provided that it is not used or sold for commercial gain. This is a general view, but in practice there are plenty of variations in the conditions of use laid down by authors. These can include such things as prohibiting modification of the software and re-distribution of the modified form without the author's consent. Others may allow commercial use with the author's permission and so on. Usually any special conditions are displayed when the program is run, or may be in an accompanying text file. Alternatively, they may be included in REM statements at the start of a Basic program.

It is worth also mentioning here, to clear up any misunderstanding, the difference between Public Domain software and Shareware. Although freely distributed like PD software, Shareware is not meant to be free. The author will request that you make some payment should you use the software regularly. This takes the

form of one single payment and for that you are entitled to support from the author in the use of the software as well as any updates to it or the documentation. The whole concept of Shareware relies on trust and a "try before you buy" policy.

In the world of the PC and computers such as the Amiga, Public Domain software abounds. The PC in particular has a wealth of Public Domain software, because of its international popularity and the fact that it has been around for some considerable time. Some of it is remarkably good and stands up well against equivalent commercial software. Now the Archimedes also has a reasonably large and ever growing pool of Public Domain software, which Beebug has had a hand in promoting via its own Public Domain Library.

So, why not the good old Beeb? It has been around for some ten years now but apart from the odd few bits that get passed around amongst friends or distributed via computer clubs, not much has been available. You can argue that perhaps the best of the home produced software ends up being offered to software houses or to magazines such as Beebug, but when you look at the age of the machine and the fact that it is a programmer's delight, there must be an awful lot of software written for it that has never been seen outside the author's own home.

With the above in mind, Alan Blundell decided that it was long overdue for someone to set up a PD library for the BBC micro, Electron and Master. He also realised, like the rest of us, that with the introduction of the Archimedes software houses and hardware developers would

BBC PD Public Domain Software Library

be producing fewer and fewer new products for Acorn's 8 bit machines.

BBC PD is the rather unimaginative but I suppose apt title of this venture into public domain for the Beeb. The catalogue of available software we received was impressive not perhaps by its size, but by virtue of the diversity of what was on offer. There is the usual crop of games and novelty programs, but along with these are some excellent utilities, ROM images, clip art, digitised sounds, music, a 6502 macro assembler and even a 'C' compiler to name but a few. Of interest to many I think, is the inclusion of some software specifically written for the Master 512 co-processor.

All the software is supplied on any of the regular format discs including 5.25" and 3.5", both ADFS and DFS. The catalogue is most informative and lets you know how many discs each program will come on (which depends on the format chosen) and, most importantly, machine compatibility is stated and whether any other hardware is required to run the particular application.

Cost of the discs is set at £1.50 for 5.25" and £1.75 for 3.5", inclusive of postage and packing. There are quantity discounts available for 10 or more and 25 or more discs. It is worth noting that 40 track disc drive owners using DFS will fare the worst as some of the programs will come on up to 4 discs and consequently cost a lot more than for say a Master owner using 80 track ADFS.

As is usual with all PD libraries, a sampler disc is available containing a selection of programs as a taster, along with a text file of the current catalogue. This is priced at £1.50 regardless of disc size. A service offered by BBC PD is the free updating of the catalogue text file provided a blank formatted disc is supplied along with return postage and packaging.

So what sort of quality of software can you expect for your money? Well, the short answer to that is 'variable' judging by the selection of software we received from BBC PD for the purposes of this feature. I was pleasantly surprised by some of what was on offer and it goes to show that there is some good software still around and available for what really amounts to a pittance. Unfortunately, it is not really within the scope of this article to give a review of any of the software available from the library, but I would encourage you to invest a mere £1.50 for the sampler disc and judge for yourselves. I don't think that you will be disappointed.

I applaud what Alan Blundell has started with BBC PD and hope that it will go from strength to strength. It can only serve to promote and maintain interest in the Beeb and confirm that there is life in the old dog yet. There is always a certain amount of criticism by some people that those who run PD libraries are making a fast buck out of other people's hard labours. I personally do not think that is so at the prices BBC PD charge. However, commercially run PD libraries undoubtedly have higher overheads than those run from home in someone's spare time. I find the BBC PD charges quite reasonable and acceptable.

As an incentive to supply any of your software for inclusion in the BBC PD library, they will supply you with software from the library in return for anything that you send. But if you do send software in, do make sure that what you send is really wholly yours. One of the biggest headaches in running a PD library is that of ascertaining whether the software sent to you truly qualifies to be Public Domain. BEEBUG magazine programs, for example, are not Public Domain, and this applies to most magazines publishing computer programs. B

An Automatic Save Utility

This excellent utility by Adrian Marshall will be an undoubted boon to all Basic programmers.

I don't know about you, but I find it all too easy to forget to make interim SAVES when typing in a program. Not so long ago I was typing in a mega-listing from a magazine (no names no pack drill) and I hadn't saved for some time when suddenly all the lights went out. No prizes for guessing who had forgotten to top the meter up! And so AUTOSAV was conceived.

This utility has been designed to provide an automatic save (using *SAVE) of a Basic program as it is being entered under the control of the Basic AUTO command. The main features are:

Regular saves at intervals defined by the user.

Easily amended defaults for all prompts.

Facility to append to an existing program or to start at a specific line number, as required.

Will work in any selected screen mode.

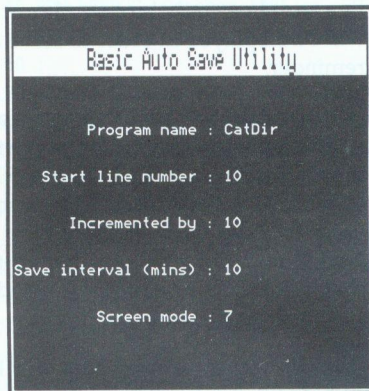
Only saves if one or more complete lines are input between intervals.

Lists the last 5 lines of an existing program to which you wish to append, thus providing visual confirmation of correct choice.

USING THE UTILITY

Type the program in and save it with a suitable name. Note that the program incorporates a movedown routine (at line 100) which automatically moves the program down in memory if PAGE is set

to more than &1300. When run you are prompted to supply various values; the program name, start line number, line number increment, save interval and screen mode. The start line number and increment are as used by the Basic AUTO command. The save interval is the time in minutes between saves, and screen mode is the mode to be used prior to issuing the Basic AUTO command.



The initial dialogue

Default values are displayed on the right of each input field (in parentheses), and a single press of the Return key is all that is needed to accept any of these. All default values for each prompt together with associated minimum, maximum, screen text and co-ordinate values are contained in DATA statements in lines 1430 - 1460 of the program listing.

Because the utility has been designed to allow new code to be appended to an existing program, it is necessary to explain the program name prompt in detail.

An Automatic Save Utility

This, in essence, is asking for the name of the file to which the Basic program is to be saved. If the file does not exist on the specified/default drive/directory, then the program will continue with the next prompt. If, however, the file does exist you are informed of this and will be given the choice of taking one of the following actions:

(I)gnore - this will ensure that the existing program will not be loaded, and line numbering will start at your specified value. Note that in this case you will be prompted to supply a start line number, and the text "Ignored" will be displayed in order to serve as a reminder.

(A)ppend - once all other prompted values have been entered, the existing program will be loaded. The last line number plus your specified increment value will be used to determine the next line number. Note that in this case you will not be prompted to supply a start line number.

(F)orce - this works in a similar fashion to Append. It will cause the existing program to be loaded, but line numbering will start at your specified value. Note that in this case you will be prompted to supply a start line number. The text "Forced" will be shown in order to serve as a reminder of this.

(Q)uit - abandon the utility.

Once all prompts have been accepted, the selected screen mode will be entered, and you will be presented with a normal AUTO prompt (and up to 5 lines of existing program and associated LIST and AUTO commands if appending). Proceed to type in your program, and

every 'n' minutes your program will be saved. When you have finished, just press Escape, and a final save will be performed and the program will terminate.

One point worth making is that the utility will only perform a save if whole line(s) are input between intervals. This means that if you find that at the start of input (and before having completed a line) you chose the wrong name to save to, you don't have to panic and hit Break or yank the disc out; just press Escape and all will be well.

HOW IT ALL WORKS

The utility uses two events, Escape condition detected (event 6) and interval timer crossing zero (event 5). The usual effect of the Escape key in Basic is disabled by enabling event 6. It is utilised in order that the Escape key code can be detected by the utility and so issue a final *SAVE. This performs all the necessary housekeeping before creating an Escape condition using OSBYTE 125, thus terminating the Basic AUTO command in the normal fashion. Event 5 is used to provide regular entry to the *SAVE routine.

The Basic section of the program is largely involved with prompting the user for values and the setting up of a suitable environment. All events apart from 5 & 6 are disabled and the user specified screen mode is selected. *KEY0 is set up with either a NEW or LOAD (depending on the outcome of the program name prompt) plus the call to the initial machine code. Finally the code for *KEY0 is inserted in the input buffer and the machine code part of the utility is called.

The initial part of the machine code preserves the value of TOP (&12 - &13) for later comparison, also PAGE (&1C - &1D) is stored for later use. The utility then format checks RAM from PAGE onwards in a similar fashion to that of Basic, additionally keeping track of the last 5 line numbers. If all is well, then the location *lflag* is examined to ascertain if an existing program is being appended to or if a start line number is being forced. In the former case, the specified increment is added to the last line number to give the start line number for the AUTO command. The latter is handled as if no program had been loaded.

The table of line numbers accumulated while checking through memory is examined, and the highest is stored in the LIST command. If the format check fails, then Basic will have reported "Bad program", so the code beeps and returns.

*KEY1 is set up with the relevant LIST and AUTO commands by the use of OSCLI. Before returning, OSBYTE 138 is used to insert the code for *KEY1 into the input buffer.

At regular intervals, the utility is entered at the code for dealing with the interval timer crossing zero (this is the code starting at the label *.timeout*). The main purpose of this code is to check for any input since it was last invoked. This is done by comparing the new value of TOP with the previously stored value. If they are different, then a *SAVE of all bytes from PAGE to TOP is performed by calling OSFILE with A=0. Having stored the new value of TOP the routine then checks to see if it was entered as a result of a type 5 event, and if so it writes to the interval timer and then does an indirect JMP via the old vector. Otherwise, it

issues an OSBYTE 125 to simulate an Escape condition and then does an indirect JMP via the old vector.

The only code left to describe is the piece that deals with the Escape key being pressed. This condition is dealt with by disabling events 5 and 6, restoring the old vector contents, and jumping to the code that does the *SAVE.

```
10 REM Program Autosave
20 REM Version B1.0
30 REM Author Adrian Marshall
40 REM BEEBUG October 1991
50 REM Program subject to copyright
60 :
100 MODE 7:IF PAGE>&1300 THEN PRINT"Re
locating":VDU21:OSCLI"K.0FORA%=0TO TOP-P
AGE STEP4:A%!&1300=A%!PAGE:NEXT:PAGE=&13
00:END|MOLD|MVDU6|MRUN|M":OSCLI"FX138 0
128":END
110 ONERROR PROCerror:END
120 tevec=&220:osbyte=&FFF4:osword=&FF
F1:oscli=&FFF7
130 ofsize=&FFDD:oswrch=&FFEE:top=&12:
page=&1C
140 temp=&70:temp1=&72:evnum=&74:lflag
=&74:temp2=&75
150 convout=&77:oldtop=&7C:oldtvec=&7E
:startline=&80
160 increment=&82:last5=&83
170 FOR Y% = 0 TO 1
180 PRINTTAB(0,Y%)CHR$134CHR$157::PRIN
TTAB(7,Y%)CHR$129CHR$141"Basic Auto Save
Utility";
190 NEXT
200 B%=-1:PRINTTAB(8,5)CHR$134;"Progra
m name :";CHR$131;
210 REPEAT
220 V%=1:INPUTTAB(24,5) "" program$
230 IF (LEN(program$) <1 OR LEN(progra
m$) >12 ) PROCferror
240 IF V%=1 PROCopenfile
250 UNTIL V%=1
```


An Automatic Save Utility

```
260 IF B% <> 0 PROCpmode
270 PROCgetinput:increment%= input%
280 PROCgetinput:interval%= input%
290 PROCgetinput:mode%=input%
300 MODE mode%:interval%=-interval%*60
00
310 PROCassemble
320 increment?0 = increment%:startline
?0 = startline% MOD 256
330 startline?1 = startline% DIV 256:i
nterval!0 = interval%
340 $name=program$:oldtvec?0=tevec?0:o
ldtvec?1=tevec?1
350 A%=13:Y%=0
360 FOR X% = 0 TO 9
370 CALL osbyte:last5?X%=0
380 NEXT
390 IF B%=1 OR B%=-1 OSCLI"K.ONEW|MCAL
L A|M" ELSE OSCLI"K.LOAD"+CHR$34+progr
am$+CHR$34+"|MCALL A|M"
400 IF B%=-1 B%=1
410 lflag?0=B%:A%=start
420 *FX14,5
430 *FX14,6
440 *FX138,0,128
450 END
460 :
1000 DEF PROCgetinput
1010 READ default%,text$,x%,y%,min%,max
%
1020 PRINTTAB(x%,y%) CHR$134;text$;" :
";CHR$131;" (";default%;"")
1030 REPEAT
1040 input$="" : INPUTTAB(x%+(LEN(text$)+
4),y%) "" input$
1050 IF LEN(input$) = 0 input% = default
t% ELSE input%=VAL(input$)
1060 IF LEN(input$)=0input$=STR$(default
t%)+ " " ELSE input$=input$+" "
1070 PRINTTAB(x%+LEN(text$)+4,y%);LEFT$(
input$,5)
1080 IF input% <= max% AND input% >= mi
n% V%=1 ELSE PRINTTAB(10,y%+2) "Value out
of range";:VDU7:V%=0
1090 UNTIL V%=1
```

```
1100 PRINTTAB(0,y%+2);SPC(40):PRINTTAB(
x%+LEN(text$)+4,y%);
1110 PRINTTAB(x%+LEN(text$)+10,y%);"
";
1120 ENDPROC
1130 :
1140 DEF PROCfnerror
1150 V%=0:PRINTTAB(12,7);"Invalid Filen
ame":VDU7
1160 ENDPROC
1170 :
1180 DEF PROCerror
1190 REPORT:PRINT" at line ";ERL
1200 ENDPROC
1210 :
1220 DEF PROCopenfile
1230 handle%=OPENIN(program$)
1240 IF handle% < 1 ENDPROC
1250 VDU7:PRINTTAB(0,7);"File already e
xists ! Please choose from"
1260 PRINTTAB(0,8);"(I)gnore - (A)ppend
- (F)orce - (Q)uit"
1270 REPEAT:B%=INSTR("AIFQ",CHR$(GETAND
&5F))
1280 UNTIL B%:B%=B%-1
1290 IF B% >=3 END
1300 PRINTTAB(0,7) SPC(80):CLOSE#handle%
1310 IF B% <> 0 ENDPROC
1320 READ default%,text$,x%,y%,min%,max
%
1330 PRINTTAB(x%,y%) CHR$134;text$;" :
";CHR$131;"Append":startline%=0
1340 ENDPROC
1350 :
1360 DEF PROCpmode
1370 PROCgetinput:startline%=input%
1380 IF B% < 1 ENDPROC
1390 PRINTTAB(x%+LEN(text$)+11,y%);
1400 IF B%=1 PRINT"Ignored" ELSE PRINT
"Forced"
1410 ENDPROC
1420 :
1430 DATA 10,Start line number,3,8,0,32
767
1440 DATA 10,Incremented by,6,11,1,255
```



```

1450 DATA 10,Save interval (mins),0,14,
1,15
1460 DATA 7,Screen mode,9,17,0,135
1470 :
1480 DEF PROCAssemble
1490 FOR pass% = 0 TO 2 STEP 2
1500 P%=&900:[OPT pass%:.save:EQUW name
1510 .loadadd:EQUW 0:EQUW &FFFF:.exadd
1520 EQUW 0:EQUW &FFFF:.startadd:EQUW 0
1530 EQUW &FFFF:.endadd:EQUW 0
1540 EQUW &FFFF:.name
1550 EQUW " " :EQUW &0D
1560 .tenhi:EQUW 0:EQUW 0:EQUW 0
1570 EQUW 3:EQUW &27:.tenlo:EQUW 1
1580 EQUW 10:EQUW 100:EQUW &E8:EQUW &10
1590 .auto:EQUW "K.1CLS|ML"::.listfrom
1600 EQUW " ,|MAU." :.autosline
1610 EQUW " ," :.autoinc
1620 EQUW " |M" :EQUW &0D:.interval
1630 EQUW 0:EQUW &FF:.start:LDX #1:.L5
1640 LDA page,X:STA temp,X
1650 STA loadadd,X:STA exadd,X:DEX
1660 BPL L5:LDA top:STA oldtop
1670 LDA top+1:STA oldtop+1:LDY #0:.L10
1680 LDA (temp),Y:CMP #&0D:BEQ L20
1690 JSR reset:LDA #7:JMP oswrch:L20
1700 JSR inctempby1:LDA (temp),Y
1710 CMP #&FF:BEQ L40:.L30
1720 LDX last5+2,Y:STX last5,Y:INY
1730 CPY #8:BNE L30:STA last5+9
1740 JSR inctempby1:LDY #0:LDA (temp),Y
1750 STA last5+8:JSR inctempby1
1760 LDA (temp),Y:SEC:SBC #3
1770 JSR inctempbyn:JMP L10:.L40
1780 LDA lflag:BNE L50:LDA last5+8:CLC
1790 ADC increment:STA startline
1800 LDA last5+9:ADC #0:STA startline+1
1810 .L50:LDA #timevent MOD 256
1820 STA tevec:LDA #timevent DIV 256
1830 STA tevec+1:LDX #0:.L80
1840 LDA last5,X:BNE L85:INX:INX
1850 CPX #10:BNE L80:BEQ L90:.L85
1860 LDY last5+1,X:LDA last5,X:TAX
1870 LDA #listfrom MOD 256:STA temp1
1880 LDA #listfrom DIV 256:JSR conv

```

```

1890 .L90:LDX startline:LDY startline+1
1900 LDA #autosline MOD 256:STA temp1
1910 LDA #autosline DIV 256:JSR conv
1920 LDX increment:LDY #0
1930 LDA #autoinc MOD 256:STA temp1
1940 LDA #autoinc DIV 256:JSR conv
1950 LDX #auto MOD 256
1960 LDY #auto DIV 256:JSR oscli
1970 LDA #138:LDX #0:LDY #129
1980 JSR osbyte:JSR timer:RTS:.timevent
1990 PHP:PHA:STA evnum:TXA:PHA:TYA:PHA
2000 .L200:LDA top:CMP oldtop:BNE L230
2010 LDA top+1:CMP oldtop+1:BEQ L240
2020 .L230:LDA top:STA endadd
2030 STA oldtop:LDA top+1:STA endadd+1
2040 STA oldtop+1:LDA page:STA startadd
2050 LDA page+1:STA startadd+1:LDA #0
2060 LDX #save MOD 256
2070 LDY #save DIV 256:JSR osfile:.L240
2080 LDA evnum:CMP #5:BEQ L250
2090 JSR reset:LDA #125:JSR osbyte
2100 JMP L260:.L250:JSR timer:.L260:PLA
2110 TAY:PLA:TAX:PLA:PLP:JMP (oldtvec)
2120 .timer:LDX #interval MOD 256
2130 LDY #interval DIV 256:LDA #4
2140 JSR osword:RTS:.reset:LDA #13
2150 LDX #5:LDY #0:JSR osbyte:LDX #6
2160 LDY #0:JSR osbyte:LDA oldtvec
2170 STA tevec:LDA oldtvec+1
2180 STA tevec+1:RTS:.conv:STX temp2
2190 STY temp2+1:STA temp1+1:CLC:PHP
2200 LDY #4:.con10:LDX #&30:.con20:SEC
2210 LDA temp2:SBC tenlo,Y:PHA
2220 LDA temp2+1:SBC tenhi,Y:BCC con30
2230 STA temp2+1:PLA:STA temp2:INX
2240 BCS con20:.con30:PLA:TXA:CMP #&30
2250 BEQ con40:PLP:SEC:BCS con50:.con40
2260 PLP:BCS con50:PHP:CPY #0:BNE con60
2270 PLP:.con50:PHP:STY temp:LDY #0
2280 STA (temp1),Y:INC temp1:LDY temp
2290 .con60:DEY:BPL con10:PLP:RTS
2300 .inctempby1:LDA #1:.inctempbyn:CLC
2310 ADC temp:STA temp:BCC inc10
2320 INC temp+1:.inc10:RTS
2330 J:NEXT:ENDPROC

```

B

Master MOS DFS Bugs

Andrew Benham offers a solution to the bugs which have been reported recently.

Recent issues of BEEBUG have highlighted some bugs in DFS 2.45 as supplied in the new Master MOS ROM. Two in particular have caused concern: the OSWORD &7D/&7E problem (see DFS OSWORD Bug in Vol.10 No.4), and the error generated when switching from ADFS to DFS (see Postbag, Vol.10 No.3). It is possible, however, to modify the DFS ROM image to fix these two problems, as described in this article. First of all I will describe the nature of the problems, and then discuss how the modification can be achieved.

OSWORD &7D/&7E

The OSWORD problem is caused by a routine at &A222, which checks that the DFS is the current filing system, overwriting the Y register which holds the OSWORD number. By making a couple of alterations it is possible to save a few bytes of code, thus making space to allow the Y register to be reloaded. Alan Wrigley's program in listing 1 will make these changes automatically and save the modified ROM as DFSImage.

ADFS TO DFS

The ADFS/DFS problem is slightly more complex but requires less modification to the code. When the ADFS floppy disc code claims the NMI handler, it stores the filing system number of the previous owner in location &C2DE. If after power-up the ADFS accesses a floppy disc before the DFS has been entered, then in a machine without any other NMI user, location &C2DE will have &FF stored in it. If however the computer is fitted with Econet and starts up with this as the default filing system, then on switching to ADFS this location will contain 5 (i.e. the network filing system number).

When the DFS is subsequently entered, locations &C2DE - &C2E1 are used as flag bytes, one for each of the 4 possible DFS drives. For some reason, &C2DE is not initialised to zero as the other three are, and as was the case in earlier versions of the 1770 DFS. If these flag bytes have bit 7 set, the drive concerned behaves as if a *Drive <n> 40 had been issued: the drive head is double-stepped and write operations are trapped. This causes the error reported in previous issues, as a result of the value of &FF being present in &C2DE. It also explains why BEEBUG technical staff were unable to reproduce the error, since their machines were fitted with Econet interfaces.

To fix the fault, the location used by the ADFS to store the number of the previous NMI owner can be altered to &C2E6, since locations &C2E6 - C2FF appear to be unused by either DFS or ADFS. Fortunately the ADFS floppy disc code is in the DFS ROM, and so listing 1 takes care of this alteration too.

EFFECTING THE CHANGES

The problem now arises of how the new DFS image can be implemented. I am fortunate in having Vine Micros' ROM Overlay Board, which allows me to substitute ROMs for sections of the Megabit ROM. I have simply blown a new EPROM which I have paged in instead of the MOS ROM's DFS. However, other users may not be so fortunate.

There is a further complication in that the ADFS floppy disc code is in the DFS ROM, and therefore the ADFS and DFS ROMs communicate with one another. In

order to do this they need to know which ROM socket the other occupies (see locations &BFEE onwards in the DFS and ADFS ROMs). If the ROMs are in sockets 8-&F, the socket number of the other ROM must be the current number EOR 4 (the standard MOS ROM uses 9 for DFS and &D for ADFS). If the ROMs are in sockets 0-7, the other socket number must be the current one EOR 1 (i.e. the two images must be in 0 and 1, or 2 and 3, or 4 and 5, or 6 and 7).

Whatever is done, the Megabit ROM versions of the DFS and ADFS will have to be *UNPLUGged. This reveals another problem: the documentation for the new MOS claims that *UNPLUG unplugs "all identical copies of a ROM"; In reality the ROMs are checked for being identical only over the first &400 bytes. So any copies which are placed in other ROM sockets must have a difference somewhere in the address range &8000 - &83FF. This is probably most easily done by making a small alteration to the title string (again, this is catered for in the DFS image by listing 1).

There are several solutions available. Firstly you could have a disc containing routines to:

1. Copy the DFS ROM to a sideways RAM slot, modifying the code on the way.
2. Copy the ADFS ROM to the adjacent RAM slot.
3. *UNPLUG the old DFS and ADFS.
4. Modify the configured filing system number if applicable.
5. Reboot.

This is inelegant, and not a permanent solution.

Secondly you could program a 27256 EPROM (32K) with the DFS and ADFS ROM images and insert these into one of

the 32K ROM sockets in the Master. This would mean losing 2 banks of sideways RAM which might prove unpopular.

Thirdly you could program two 27128 EPROMS (16K) with the DFS and ADFS images and insert these into a ROM cartridge. This would mean losing 2 cartridge slots, but ROM cartridges tend to be a rather under-used feature of the Master 128.

In my opinion, however, the Overlay Board approach is the neatest and least likely to cause problems. But whichever method you adopt, this will do something to dispel the disappointment many users feel at the release of a major software upgrade which contains such serious bugs.

```
10 MODE135:HIMEM=&4000
20 PRINT"Reading ROM image..."
30 FORI%=&8000 TO &BFFF
40 PRINTTAB(21,1);~I%;
50 Y%=9
60 ?&F6=I% MOD 256:??&F7=I% DIV 256
70 ?(I%-&4000)=USR(&FFB9)
80 NEXT
90 FOR I%=0 TO 2 STEP 2
100 P%=&5194
110 [OPT I%
120 STX &C7:STY &C8:LDA #&FF:STA &C282
130 ]
140 P%=&622C
150 [OPT I%
160 LDY &EF:NOP:NOP:LDX &F0
170 STX &C7:LDX &F1:STX &C8:INY
180 BPL skip:JMP &91F8
190 .skip
200 ]
210 NEXT:??&4018=&39
220 ?&7B69=&8C:??&7B6A=&E6:??&7B6B=&C2
230 ?&7B6D=&AC:??&7B6E=&E6:??&7B6F=&C2
240 PRINT""Saving modified DFS as ""
DFSImage""
250 *Save"DFSImage" 4000 8000
```

B

TexBase (2)

by Paul Pibworth

Last month I introduced you to the idea of TexBase as a system to store and retrieve text based data. If you typed in the listings given, you should have been able to enter and store text. If you did enter some text, you would have found that you could not actually do anything with it at the time. However, we will solve that problem now by moving on to the important part, the search options.

Type in listing 1, which is the search program, and save it with the name *Search*. Like the Enter program (from last month), it has to be run from the main menu program (TMenu), which sets up the VDU23 defined characters. Also like Enter, it is structured, and runs from a small loop so you should be able to follow what happens. You may like to change the lines that return you to the TMenu program, until the program has been fully tested. Replace CHAIN"TMENU" found near the end of the program with:

```
OSCLI"FX229,0":OSCLI"FX4,0":END
```

When Search is run, a brief menu is presented, with just five choices. These will be described in turn.

- 1 ENTER KEY WORDS
- 2 LOGIC = OR (Press to change)
- 3 SEARCH
- 4 START AT PAGE
- 5 EXIT

OPTION 1. ENTER KEY WORDS

You will remember from last month that words in the text could be chosen as key words. Selecting option 1 allows you to enter up to six keywords. These form the basis of the search. If none are chosen,

then a wildcard is chosen, and every page is displayed. You can change, delete, or modify your list of key words from within option 1. The program works by using the INSTR command. This means that a small "search word" can be found within larger keywords. E.g. using RUM would display a page which had CRUMB or DRUM. This can be prevented by preceding the search word with a slash, e.g. /RUM. A slash can also be included as a terminator, DRUM/ would not find DRUMS, although DRUM itself would.

```
                                > Page:1
                                From:1
line:=====
1  Last month I introduced you to the idea of TexBase as a system to store and
2  retrieve text based data. If you typed in the listings given, you should
3  have been able to enter and store text. If you did enter some text, you
4  would have found that you could not actually do anything with it at the
5  time. However, we will solve that problem now by moving on to the
6  important part, the search options.
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Using the Search program to retrieve text

OPTION 2. 'OR' AND 'AND' SEARCH LOGIC

In the latter case, each and every one of the key words must be matched. Selecting option 2 toggles between these two states.

OPTION 3. SEARCH

This option looks at each group of keywords in turn, until a match is found. When (and if) this occurs, (and it may take time - be patient) the page is then

displayed. The cursor keys with Shift, and Ctrl, can move the page up and down.

When each page is displayed, you have four choices, as shown at the base of the screen. N(next) will continue the search, and E(exit) will return you to the menu. P(rint) will allow you to print out all or some of the page - you enter the required answers in response to the questions. The questions appear on the ruler, near the top of the screen. Printing makes use of the procedure PROCcode (line 2650). This has the printing codes in it that match my printer (an Epson FX80). You will need to change these if necessary, but I think they are standard.

S(ave) enables you to save the page, as an ASCII file. If your disc is full (i.e. you have the maximum sized data file possible), you will have to change discs. The program assumes this; just answer 'Y', and ignore it if you do have room on the disc. Each time you save such text, it is given the filename "savedtx". You then need to *COPY this or *RENAME it at your convenience, or you will overwrite it in the future. If you are using a Master with an ADFS disc, you will have to introduce an OSCLI" MOUNT" after each disc change.

OPTION 4. START AT PAGE

This enables you to start the search some way through the file, if you know for example what page the data is actually in. This may seem to nullify the whole point of keyword searching, but it can prove useful (and quicker) if returning to the same information, of if you have a separate index (more later).

THE DATA FORMAT EXPLAINED

That has covered the searching of the data. Now we will look at the way the programs work. You must remember that

each text entry is stored inside the random access file, "text", which has already been set up. Each entry, or page, begins with 154 bytes which contain the following information. There is the page number, the number of lines in that piece of stored text, the keywords, (as one string, starting and ending with a slash), and the title. Each line on the screen is a 75 character entry in a string array. When the entry is saved, each line takes up 77 characters on the disc. The length of each entry, or page, is thus a multiple of 77. Any blank lines that remain on the screen after your last entered line are not stored.

The first two sectors of the data file are used as an index, where byte 0 contains the number of pages that have been saved. The modified address of each page is stored as a 2 byte number. Bytes 1 and 2 give the address of page 2, bytes 3 and 4 give page 3 etc. Page one starts right after the index, so its address does not need to be stored. To find the exact address, each 2 byte number is then multiplied by 77, and &200 is added. Thus, if a page number is known, its address can be calculated, and the page loaded by using the OPENUP command, and then PTR#Z%=address.

Conversely, by stepping through the addresses, each entry can be located and the keyword string loaded and compared, until a match is found. When creating a new entry, the address of the last page is used to find that last page, read its length, and thus put the new entry on the end. The index bytes are then updated.

When using the Enter program, the next available position is calculated prior to displaying the blank screen ready to receive your entry. Originally, I wanted the number of entries (pages) to be

unlimited. An index speeds up search, but it limits the number of entries. Two hundred and fifty 10 line entries would need $(250 \times (10+2) \times 77) + 512$ bytes = 231512 bytes. I compared this to the 102400 bytes on a 40 track DFS disc, and decided to compromise. Thus there is a limit of 250 pages per disc. However, one could modify the program to have a choice of two or more data files.

CREATING A DATA FILE

This leads on to the creation of the data file, "text" - we will assume that there is only one. In the first article there was a one-liner to set up a short file, to use for test purposes. Next month I will include a program to do this automatically, and another to remove a page. However, I can show you how to do it yourself, and set up the maximum data file possible. Once you have all the programs you need on the disc, (and that should include next months program, *Delete*), you must *COMPACT, and type *INFO *.* This shows a series of lines such as:

```
Delete 001400 008023 000829 031
```

Assuming that this is the top row displayed, it is the last file. The third group of digits gives the program length, in hexadecimal (in the above example it is &829). This must be converted into sectors, so using the numbers above, you would type:

```
PRINT&829/256
```

This will give 8.160156. Round this up to 9, the number of sectors used. It started at &31, i.e. sector 49. That means the next available sector is at $9+49=58$. Take this from 400 (or 800 for an 80 track disk), you now have 342 sectors. Subtract 2 sectors, convert to bytes and divide by 77, and round down to an integer (i.e. $\text{PRINT INT}(340 \times 256 / 77)$). This tells you

how many lines can be stored, e.g. =1130 lines. Now multiply by 77, and add 512, (=87522), and use this number in the one liner in the last article. This will give the maximum size, being based on the unused disc space. You can of course make it smaller, but it must be a multiple of 77, plus 512.

```
10 REM Program Search
20 REM BEEBUG October 1991
30 REM Subject to copyright
100 CLOSE#0
110 ONERRORGOTO2930
120 DIMK$(7):K%=0:L%=-1:lgc$="OR":O%=-
3:pr$="PRESS RETURN"
130 MODE 3
140 DIMA$(50):PROCblk
150 *FX4,1
160 REPEAT:PROCmain:UNTILM%=53:
170 CLOSE#0
180 END
1000 REM Program Search
1010 REM BEEBUG October 1991
1020 REM Subject to copyright
1030 CLOSE#0
1040 ONERRORGOTO2930
1050 DIMK$(7):K%=0:L%=-1:lgc$="OR":O%=-
3:pr$="PRESS RETURN"
1060 MODE 3
1070 DIMA$(50):PROCblk
1080 *FX4,1
1090 REPEAT:PROCmain:UNTILM%=53:
1100 CLOSE#0
1110 END
1120 :
1130 DEFPROCmain:C%=POS:D%=VPOS:VDU22,7
1140 PRINTTAB(15,3)"M E N U""1:ENTER
KEY WORDS""2:LOGIC = "lgc$" (press to c
change)""3:SEARCH""4:START AT PAGE""5:
EXIT"
1150 REPEAT:M%=GET:UNTIL(M%>48ANDM%<54)
1160 IF M%=49 PROCkeyw
1170 IF M%=50 lgc$=FNlog
1180 IF M%=51 VDU22,3:PROCindex:PROCser
(&200)
1190 IF M%=52 PROCgoto:IFfindp%>0 THEN
VDU22,3:PROCser(P%)
```



```

1200 IF M%=53 PROCend
1210 ENDPROC
1220 DEFPROCkeyw
1230 CLS:IFK%=0 PROCentkw:ENDPROC
1240 PRINT"Your current keywords are;"
1250 PRINT
1260 FORN%=1TOK%:PRINT;N%;" ";k$(N%):NE
XT
1270 PRINT
1280 PRINT"(R)edo, (D)elete, (A)dd,""(
M)odify, (O)K"
1290 REPEAT:R$=CHR$(GETAND95):UNTILINSTR("ARDMO",R$)>0
1300 IFR$="O" ENDPROC
1310 IFR$="R"PROCentkw:ENDPROC
1320 IFR$="A" AND K%<6 K%=K%+1:INPUT"Enter: "k$(K%):ENDPROC
1330 INPUT"Which one? "R%:IFR%>K% R%=K%
1340 INPUT"Enter (null to delete): "k$(R%)
1350 IFk$(R%)=""THEN FORN%=R% TO K%:k$(N%)=k$(N%+1):NEXT:K%=K%-1
1360 PROCkeyw
1370 ENDPROC
1380 DEFPROCentkw
1390 INPUT"How many key words? (0 - 6)"
1400 REPEAT:k$=GET$:UNTILINSTR("0123456",k$)>0
1410 K%=VALk$:IFK%=0 ENDPROC
1420 PRINT:FORN%=1 TO K%
1430 *FX202,32
1440 PRINT"Enter no ";N%;:INPUT" "k$(N%)
)
1450 NEXT
1460 ENDPROC
1470 DEFfnlog
1480 L%=L%*-1:IFL%=-1 THEN="OR"
1490 ="AND"
1500 DEFPROCgoto
1510 PRINT"if you want to start the"" search at a given page, then""enter that page, (else return)"
1520 INPUT""findp%
1530 IFfindp%=0:ENDPROC
1540 Z%=OPENUP"text"
1550 PTR#Z%=0:G%=BGET#Z%
1560 IFfindp%>G% PRINT"NO SUCH PAGE!":CLOSE#Z%:A%=INKEY(200):findp%=0:ENDPROC

```

```

1570 IFfindp%=1 P%=&200:ENDPROC
1580 PTR#Z%=findp%*2-3:ptr%=BGET#Z%
1590 ptr%=ptr%*256+BGET#Z%
1600 P%=&200+77*ptr%
1610 ENDPROC
1620 DEFPROCindex
1630 Z%=OPENUP"text"
1640 PTR#Z%=0:G%=BGET#Z%:REM no of pages
1650 ENDPROC
1660 DEFPROCser(P%)
1670 IFK%=0 PRINTTAB(20,5)"YOU HAVE NOT ENTERED KEYWORDS"TAB(20)"THIS WILL DISPLAY EVERY PAGE"TAB(20)"PRESS C TO CONTINUE THE SEARCH":A%=GET AND 95:IF A%<>67:CLOSE#Z%:ENDPROC
1680 B%=0:F%=0:REM found flag
1690 REPEAT
1700 PTR#Z%=P%:INPUT#Z%,pg%
1710 INPUT#Z%,lines%:INPUT#Z%,kw$
1720 INPUT#Z%,ti$:PROCcomp
1730 UNTILB%=ASC"E" OR pg%=G%
1740 CLOSE#0
1750 IFF%=0PRINTTAB(20,10)"NOT FOUND!":F%=INKEY(200)
1760 ENDPROC
1770 DEFPROCcomp
1780 IFK%=0:PROCread:ENDPROC
1790 LOCALyes%:yes%=0
1800 FORN%=1TOK%
1810 IFINSTR(kw$,k$(N%))>0 yes%=yes%+1
1820 NEXT
1830 IF L%=1 AND yes%=K% PROCread:ENDPROC
)
1840 IF L%=-1 AND yes%>0 PROCread:ENDPROC
)
1850 PROCskip
1860 ENDPROC
1870 DEFPROCread
1880 P%=P%+154:PTR#Z%=P%
1890 FORN%=0TOLines%-1
1900 INPUT#Z%,A$(N%)
1910 NEXT
1920 FORN%=lines% TO49
1930 A$(N%)=STRING$(75," ")
1940 NEXT
1950 P%=P%+(lines%*77):REM ready for next read

```



```

1960 O%=-3:CLS:PROCdis
1970 VDU23,1,1;0;0;0;
1980 F%=1
1990 REPEAT
2000 *FX202,32
2010 B%=GET:PROCkey
2020 UNTILB%=ASC"E" OR B%=ASC"N"
2030 ENDPROC
2040 DEFPROCskip
2050 P%=P%+(lines%+2)*77)
2060 ENDPROC
2070 DEFFNnum(a$):PRINTTAB(0,2)a$;" =
====";:INPUTTAB(LENA$+1,2)""a%:=a%
2080 DEFFNques(a$):PRINTTAB(0,2)a$;" ";
:REPEAT:R$=GET$:UNTILINSTR("YN",R$)>0:PR
OChd:=R$
2090 DEFFNcol(A)
2100 =LEFT$(STR$(A)+" ",5)
2110 DEFPROCblk:FORA%=0TO50:A$(A%)=STRI
NG$(75," ") :NEXT:ENDPROC
2120 DEFPROCkey
2130 IFINKEY(-2)PROCctr:ENDPROC
2140 IFB%=138PROCup
2150 IFB%=139PROCdn
2160 IFB%=ASC"P" PROCprint
2170 IFB%=ASC"S" PROCsave
2180 ENDPROC
2190 DEFPROCctr
2200 IFB%=138 O%=27:PROCdis:ELSEIFB%=13
9 O%=-3:PROCdis
2210 B%=0:ENDPROC
2220 DEFPROCdis
2230 LOCAL A:PROChd:PROCbot:PRINTTAB(0,
3);:FORA=0%+3TOO%+22:PRINTFNcol(A+1);A$(
A);:NEXT:ENDPROC
2240 DEFPROChd
2250 PRINTTAB(0,0)"KEYWDS <";SPC60;TAB(
8,0);kw$;TAB(68,0)"> Page:";pg$;TAB(9,1)
;"TITLE :";ti$;TAB(70,1)"From:";G$;
2260 PRINTTAB(0,2)"line:";TAB(5,2)STRIN
G$(15,"====:");
2270 ENDPROC
2280 DEFPROCbot
2290 PRINTTAB(0,23)STRING$(79,"=");
2300 IFO%=27PRINTTAB(30,23)"BOTTOM OF
PAGE";
2310 PRINTTAB(10,24)"(N)ext (E)xit (
P)rint (S)ave";

```

```

2320 ENDPROC
2330 DEFPROCup
2340 O%=O%+10:IFO%>27 O%=27
2350 LOCAL A:PRINTTAB(0,3);:FORA=0%+3TO
O%+22:PRINTFNcol(A+1);A$(A);:NEXT
2360 ENDPROC
2370 DEFPROCdn
2380 O%=O%-10:IFO%<-2 O%=-3
2390 LOCAL A:PRINTTAB(0,3);:FORA=0%+3TO
O%+22:PRINTFNcol(A+1);A$(A);:NEXT
2400 ENDPROC
2410 DEFPROCprint
2420 IFFNques("Confirm printout Y/N")="
N":ENDPROC:ELSEq$=FNques("Print out all?
Y/N ")
2430 C%=1:D%=lines%
2440 IFq$="N"C%=FNnum("Top line"):D%=FN
num("Bottom line (0=abort)"):IFC%*D%=0:
PROChd:ENDPROC
2450 PROChd
2460 *FX3,10
2470 IFti$<>" PRINTti$:PRINT
2480 IFD%>lines% D%=lines%
2490 FORN%=C%-1 TO D%-1
2500 PROCpr2(A$(N%))
2510 NEXT
2520 *FX3,0
2530 ENDPROC
2540 DEFPROCpr2(a$)
2550 S%=0:pos%=76
2560 REPEAT
2570 pos%=pos%-1
2580 UNTILMID$(a$,pos%,1)<>" OR pos%=1
2590 a$=LEFT$(a$,pos%)
2600 FORI%=1TOpos%
2610 A%=ASC MID$(a$,I%,1)
2620 PROCcode
2630 NEXT:VDU13
2640 ENDPROC
2650 DEFPROCcode
2660 IFA%=96 A%=35
2670 IFA%<127 VDUA%:IFS%>0 VDU27,84:S%=
0
2680 IFA%<127 ENDPROC
2690 IFA%=240 VDU27,52
2700 IFA%=241 VDU27,45,1
2710 IFA%=242 VDU27,83,1:S%=1

```

Continued on page 24

Wordwise User's Notebook

Wordwise Plus Teletext Effects (1)

by Chris Robbins

One of many attractive features of the Wordwise Plus (WW+) language is its ability to use special screen effects to enhance the operation and presentation of text processing programs.

SECURITY

Take for example the question of security. Using the Teletext conceal code it's possible to enter a password, or perhaps some confidential information into a document, without any danger of it being noticed by prying eyes. The following is one way of using it to provide a simple deterrent to curious minds and light fingers.

```
CLS
REM Disable ESCAPE
OSCLI("FX200,1")

PRINT "Enter Password ";
VDU152
P$=GLK$
IF P$<> "Fred" THEN GOTO wrong-password
```

```
PRINT
PRINT "Password correct"
REM Enable ESCAPE
OSCLI("FX200,0")
END
```

```
.wrong-password
PRINT
PRINT "Wrong Password"
REPEAT
VDU7
N%=GET
UNTIL 0<>0
END
```

VDU152 ensures that what's typed in isn't displayed, although as it stands, the code is far from foolproof. However,

getting round it by means of the Escape key is blocked by the initial OSCLI call to disable it. And if any attempt is made to guess the password, an alarm sounds and the keyboard is effectively locked (apart from the Break key) by the REPEAT-UNTIL construct.

Security could be improved still further using a 'load and go' approach that bypasses menu mode to get the application up and running. Additionally, a self-destruct mechanism could be employed that wipes the relevant segments and text areas in the event of an incorrect password.

An even more effective, although rather drastic approach, is to wipe *all* memory, especially if Break is pressed in an attempt to get around the security mechanism. And, but I must stop there as I'm straying rather from the main topic.

COLOUR ON DISPLAY

Back to Teletext. An important consideration when devising displays is the question of colour. WW+ usually displays text as white on black, but this can be changed; certainly as far as messages presented by segment programs are concerned. The following code might be used in order to enter a name into a standard letter, say:

```
PRINT "Enter Name ";
N$=GLK$
```

Although adequate, it could be made more interesting by displaying the prompt in a different colour. For example, preceding the PRINT command by VDU131 (note - WW+

Wordwise User's Notebook

allows only one command per line) ensures that both prompt and user response are displayed in yellow.

A further 'improvement' might be the use of a different background colour, blue say, by using instead VDU132,157,131. The first part of the command (VDU132,157) sets the background to blue, whilst the last part (,131) sets yellow text. In an even bolder attempt to catch the user's eye, this could be made to flash by adding a further code to give VDU132,157,131,136.

Another variation might be to set up an entirely different text/background colour combination for the screen area in which the user response is displayed. Putting all these features together could give WW+ code something like this:

```
CLS
VDU132,157,131,136
PRINT"Enter Name ";
VDU131,157,132,137
N$=GLK$
```

This produces a flashing yellow prompt on a blue background, whilst user input is displayed as blue text on a yellow background; the final 137 in the second VDU command ensuring that users don't type anything flashy! The initial CLS is simply there to clear the screen, and home the cursor to the top left-hand corner before the prompt is displayed.

Ideas such as this can prove particularly useful when it comes to displaying warnings or error messages, especially if they're built into standard 'reusable' routines, as in the following example:

```
M$="Don't Panic"
REM Red background
B%=129
REM Yellow text
T%=131
REM Flashing text required
```

```
F%=TRUE
PROCdisplay-message
END
```

```
.display-message
REM M$ passes message to be displayed
REM F% Flash flag. TRUE if required
REM B% background colour
REM T% text colour
```

```
CLS
IF B%<>0 THEN VDU B%,157
IF T%<>0 THEN VDU T%
IF F%=TRUE THEN VDU 136
PRINT M$;
IF B%<>0 THEN PRINT " ";
IF T%<>0 THEN PRINT " ";
IF F%<>0 THEN PRINT " ";
REM Restore black background
VDU 156
```

```
REM Hide cursor
VDU 23;11,0;0;0;0;
```

```
ENDPROC
```

The first few lines in the above program set up the variables to be used by the procedure *display-message*, whilst display of "Don't Panic" is invoked by a call to the procedure, to give flashing yellow text on a red background at the top left-hand corner of the screen.

There is obviously considerably more code here than absolutely necessary to display the message; the REMs for instance perform no useful run-time function, whilst the active code could be reduced to:

```
CLS
VDU129,157,131,136
PRINT "Don't Panic";
VDU156
VDU23;11,0;0;0;0;
```

and still produce the same effect. However, the object of the exercise has

been to demonstrate possibilities, not minimal coding.

POSITIONING TEXT

Text display is open to all sorts of variations. The position on the screen for instance; omitting the CLS will cause the message to be displayed at the current cursor position, the standard place being somewhere on a line halfway down the screen. Alternatively VDU31,x,y (x=horizontal position, y=vertical) can be used to place it anywhere on the screen.

For example, VDU31,21,0 will reposition the text cursor so that the flashing message appears at the top right hand corner of the screen (full details on cursor control are given in the standard BBC User Guide). Note that using VDU31 does not interfere with WW+ cursor positioning. Incidentally, if you need to know the actual current horizontal position of the WW+ cursor then use ?126 (e.g. P%=?126 places the x position in the integer variable P%).

Yet another advantage (at times) of using VDU31 instead of CLS, almost goes without saying - the screen isn't cleared; the cursor could be positioned so that a suitable message is displayed in a convenient area of the screen, with the cursor then repositioned so that it indicates the point in text or screen display to which the message relates. For example:

```
VDU31,8,24
VDU129,157,131,136
PRINT "Duplicate Name  ";
VDU156
VDU31,?126,13
```

This produces the flashing message "Duplicate Name" in yellow on red in the centre of the bottom line of the screen, then repositions the cursor at the correct current WW+ position (to change

the actual WW+ cursor position use the WW+ CURSOR command). Although real applications of this idea are many, the example given, could, for instance, be used in a WW+ program that checks mailing lists for duplicate entries.

SPECIAL EFFECTS

Double height messages can be used to give even greater impact. For example:

```
CLS
VDU28,0,13,39,11
REPEAT
REM Yellow background blue text
    VDU131,157,132
REM Double height
    VDU141
REM Flash on
    VDU136
    PRINT "          FILE NOT FOUND"
TIMES 2
REM Hide cursor
VDU23;11,0;0;0;0
```

conveys an unmistakable and unavoidable message in flashing blue double height letters on a yellow background. The initial VDU28 sets up a text window halfway down the screen, in which the message is displayed. REMs have been included in the above simply to explain the other VDU commands. These separate commands can of course be combined to give one command i.e. VDU131,157,132,141,136.

One of the problems of using a repeat loop in the generation of double height characters is the inability to reset background colours. This minor difficulty can be overcome simply by using two lots of code. This approach has the added bonus that by so doing it's possible to produce all sorts of extra and interesting effects. For example, the following WW+ code produces a centralised 'tidied up' background for the flashing blue message, with a yellow

Wordwise User's Notebook

background for the top half of the letters and a red background for the bottom.

```
CLS
REM Set up central text window
VDU28,6,13,31,11
VDU141
REM Yellow background blue text
VDU131,157,132
VDU136
PRINT "FILE NOT FOUND  ";
REM Reset background colour
VDU156
PRINT
VDU141
REM Red background blue text
VDU129,157,132
VDU136
PRINT "FILE NOT FOUND  ";
VDU156
VDU23;11,0;0;0;0
```

One of many possible variations on this theme is different colours for the two halves of the letters; blue and red on a common yellow background is particularly effective. Yet another interesting effect is to flash only the top or bottom half of the message by omitting the appropriate VDU136.

It's even possible to use the Teletext graphics characters from within WW+. The following displays the characters produced by values 32 to 63:

```
CLS
PRINT
I%=32
VDU147
REPEAT
VDU I%
I%=I%+1
UNTIL I%>63
```


The VDU147 switches on yellow graphics until the end of the current line (it has to be switched on for each line). The result does look rather confusing; it's easier to grasp the actual shapes generated, by displaying them individually, or by turning on separated graphics using VDU154 prior to the REPEAT command.

Although the use of graphics from within WW+ gives a powerful screen presentation capability, it requires a deal of artistic and creative ability to produce satisfactory images. But as demonstrated by 'real' systems, results can be impressive.

(to be continued next month) 

TeXBase (continued from page 20)

```
2720 IFA%=243 VDU27,83,0:S%=1
2730 IFA%=244 VDU27,69
2740 IFA%=245 VDU27,53,27,45,0,27,70
2750 ENDPROC
2760 DEFPROCsave
2770 IFFNques("Confirm saving to another disc Y/N")="N":ENDPROC:ELSEq$=FNques("Save all? Y/N ")
2780 C%=1:D%=lines%
2790 IFq$="N"C%=FNnum("Top line"):D%=FNnum("Bottom line (0=abort)":IFC%*D%=0:PROChd:ENDPROC
2800 PROChd
2810 IFFNques("HAVE YOU CHANGED DISC?")="N":PROChd:ENDPROC
2820 Y%=OPENOUT"savedtx"
```

```
2830 IFTi$<>" FOR S%=1TOLENTI$:BPUT#Y%,ASCMIID$(ti$,S%,1):NEXT:BPUT#Y%,13
2840 IFD%>lines% D%=lines%
2850 FORN%=C%-1 TOD%-1
2860 FOR S%=1TO75:BPUT#Y%,ASCMIID$(A$(N%),S%,1):NEXT:BPUT#Y%,13
2870 NEXT
2880 CLOSE#Y%
2890 REPEAT:UNTILFNques("HAVE YOU RETURNED TO ORIG DISC?")="Y"
2900 ENDPROC
2910 DEFPROCend:CHAIN"TMENU"
2920 REM ERROR TRAP
2930 VDU3:OSCLI"*FX3,0":CLOSE#0:CLS:PRINTERR;" at ";ERL:PRINTPr$:B%=GET:IFB%=13 GOTO1090 ELSE CHAIN"TMENU" 
```


Simulation Modelling

by Mike Williams

Last month, in a new series of the BEEBUG Workshop, I discussed random sampling. That Workshop showed how routines could be written to obtain a random sample from either a negative exponential or a normal distribution, two of the commonest distributions to be found in real life. The reason for covering random sampling was to provide ourselves with an essential tool for an investigation of *simulation modelling*. This is what I want to make a start on this month, and we shall be utilising the work covered last time.

By considering a number of examples, the intention is to show how a simulation program can be constructed. Since we will be using Basic, rather than a specialised simulation language, any simulation model will have to be individually programmed. What I hope to do here is to present the basic principles and identify an outline program structure which will suit a wide range of such applications. It is also likely that any procedures and functions which we

write will be suitable (in modified form) for other similar programs.

Essentially, we are dealing with queuing systems of one kind or another, and we have all experienced examples of this - petrol stations and supermarket checkouts are just two of the more obvious examples. One fundamental feature of such systems is that they are *time-based*. Any simulation model will involve the notion of some form of clock mechanism by which time can be advanced, and the model will need to be run over some simulated period of time.

There are two other terms which we need to get to grips with. These are *activities* and *events*. An activity is any task which occupies time - it has a duration. Waiting in a queue is an activity (believe it or not); so is being served. Events are points in time at which the status of the model changes - a petrol pump becomes free; a car arrives; etc.

In general, a simulation model will be either *event-centred* or *activity-centred*, and I want to take a simple example and show how each approach can be applied to it. If you are wondering where our random sampling comes in, that's easy. The timing of most events is determined by random sampling from a suitable distribution. For example, the times at which people (say) join a queue can be determined by random sampling; the time taken to serve a customer can be determined by random sampling; and so on.

In practice, we can either monitor a genuine situation, recording times etc, to build up a distribution which can then be incorporated in our model, or we can

Workshop - Simulation Modelling

use a theoretical distribution (like the normal distribution discussed last month) and sample from that with appropriate mean and standard deviation. We can also envisage a *calendar* of future events. The main loop of the program will advance the clock to the time of the next event. All the changes to the model which then arise can be implemented, and the time advanced once more to the next event.

SIMPLE EXAMPLE

Let us look at the modelling of a simple and slightly idealised queuing system. At intervals, a customer arrives to be served by a single *server*. In such a system we can identify three activities. Each activity comprises a *test head* (to determine if that activity is to be performed), and an *action*. In our simple model, the following activities can be identified:

waiting

test: Has a customer arrived?

being served

test: Is there a waiting customer *and* an idle server?

idling

test: Has a service finished?

We can thus represent the program in outline as follows:

Initialise model

Set arrival time of first customer

REPEAT

IF a customer has arrived THEN
 Increment customer queue length
 Set next arrival time

IF queue>0 AND server idle THEN
 Set server busy
 Set end-service time
 Decrement queue length

IF end-service time reached THEN

Set server idle

Advance clock to time of next event

UNTIL simulation period complete

A program implementing this is given in listing 1. The model is initialised by PROCinit (see line 1040 onwards): the clock is set to zero, as is initial queue length. The model uses two *time cells*, represented by the variables *arrival* and *end_service*. These are initially set to zero, but subsequently contain the times of the next arrival, and the next end-of-service time. The clock is always advanced to whichever of these occurs first. Note that all times are stored as absolute clock times (rather than relative to the current clock time). However, this is not important: just decide on one approach or the other.

The pseudo program given above refers to whether the server is idle or not. This is handled by the variable *server_idle* which is set to TRUE when the server is idle (its initial state), and to FALSE otherwise.

Other values are prompted for input by the user. These include the duration of the simulated period, and the mean and standard deviation of each of two distributions to be used for sampling, one for arrival times and one for service times. I have used the FNnormal6 function from last month, modified as described then to generate values related to a given mean and standard deviation. You do need to watch that sampling does not introduce negative values (unless you have found some way of making time run backwards). You might consider that sampling from a negative exponential distribution would be more appropriate (or maybe a Poisson distribution). Any random sampling

function as appropriate could be substituted for the FNnormal used in the program.

To get the model up and running, a first arrival is set up in line 140, and the clock advanced to this time. The main loop of the program then follows. In essence this simply mirrors the pseudo program given above. Line 200 advances the clock to the time of the next event, whichever is the smaller of *arrival* and *end_service*.

```
Duration: 240
Arrival times
  Mean: 5
  Standard deviation: 1.5
Service times
  Mean: 6
  Standard deviation: 1
Clock = 6           queue = 0
Clock = 11          queue = 0
Clock = 12          queue = 1
Clock = 16          queue = 0
Clock = 19          queue = 1
Clock = 23          queue = 0
Clock = 25          queue = 1
Clock = 28          queue = 0
```

Figure 1. Sample output

There are, unfortunately two complications that arise. Because of the ordering of activities it is possible for the server to be set idle just after (but at the same time as) a customer has entered the queue. This is catered for by the repeated test for queue length greater than zero in line 190. We also have to ensure that the clock really does move on by selecting the smaller of the two time cells *which is greater than the current clock time* (see line 200 again).

If you were to run the simulation as envisaged, then you would be quite disappointed. Nothing would appear to

happen. To see what is going on, line 160 has been added to the original algorithm. Each time the clock has been advanced, the program displays the current time and queue length. This gives quite an interesting and dynamic view of the queuing situation. A sample of output is shown in figure 1.

In more serious applications of simulation modelling, much more attention is paid to collecting results. For example, in this instance we would probably want to calculate the mean and standard deviation of the recorded queue lengths. Then we can start answering questions like, "What is the average queue length?", or "What is the probability of the queue exceeding a certain size?"

We also have to remember that the results of running this program depend on the sequence of random numbers generated. Each run of the program, even with the same input, will produce different results. We therefore need to run the simulation a number of times (at least 30 is usually considered necessary for statistically useful results) averaging out the results over the 30 trials. Of course, the program could be easily amended to do this automatically.

Next month's workshop will look at an event-based approach to the same simple example, and we will also be looking at some other more complex examples of queuing systems. In the meantime you might like to consider changing the model as follows. Assume that after some specified time customers stop arriving (the shop, garage or whatever closes). The simulation should now continue running until the last customer has been served, so that the additional time needed after closing can be assessed.

```

10 REM Program Activity
20 REM Version B1.0
30 REM Author Mike Williams
40 REM BEEBUG October 1991
50 REM Program subject to copyright
60 :
100 MODE131:ON ERROR GOTO 240
110 PROCTitle
120 PROCinit
130 PROCinput
140 arrival=clock+FNnormal(m1,sd1):clo
ck=arrival
150 REPEAT
160 PRINT"Clock = ";clock;TAB(20);"que
ue = ";q
170 IF clock=arrival THEN q=q+1:arriva
l=clock+FNnormal(m1,sd1)
180 IF q>0 AND server_idle THEN server
_idle=FALSE:end_service=clock+FNnormal(m
2,sd2):q=q-1
190 IF clock=end_service THEN server_i
dle=TRUE:IF q>0 THEN server_idle=FALSE:e
nd_service=clock+FNnormal(m2,sd2):q=q-1
200 IF arrival<=clock clock=end_servic
e ELSE IF end_service<=clock clock=arriv
al ELSE IF arrival<end_service THEN cloc
k=arrival ELSE clock=end_service
210 UNTIL clock>clock_end
220 END
230 :
240 IF ERR<>17 THEN MODE131:REPORT:PRI
NT" at line ";ERL
    
```

```

250 END
260 :
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
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
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)
    
```

B

CORPLAN

For serious work with Wordwise Plus

- ❖ Reviewed in BEEBUG Vol.9.No.8
- ❖ Descriptive indexing for your letters & documents
- ❖ Your own library of layout forms, letterheads etc.
- ❖ Automatic import of addresses, references, dates etc.
- ❖ CORPLAN does the layout, you just type the text!
- ❖ Resident utilities for mailmerge, label printing etc.
- ❖ For B, B+ & Master. Needs discs & Wordwise Plus
- ❖ Many special features - see free information sheet
- ❖ Pack contains disc, tutorial manual, keystrip etc.
- ❖ Price £19.50, post free UK, 14 day refund



CORPLAN Computer Systems

Three Gables, 7a Talbots Drive, Maidenhead,
Berks, SL6 4LZ. Phone or Fax (0628) 24591

*Wish something new was happening for
your BBC Micro, Master or Electron?
Something is!*

BBC Public Domain software library has over
20Mb of software available!

Send £1.50 for catalogue and sampler disc to:

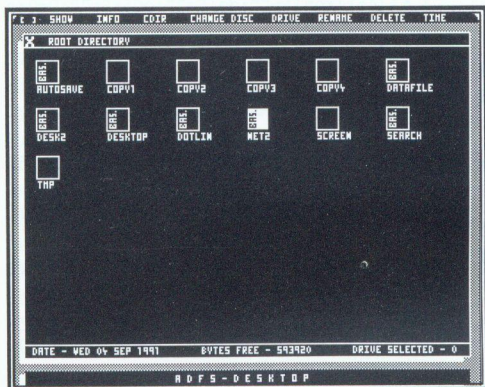
**BBC PD, 18 Carlton Close, Blackrod,
Bolton, BL6 5DL.**

Make cheques payable to;
'A Blundell' or send an A5 s.a.e for more details.
(Please state disc size and format)

ADFS Desktop Enhancements

Terry Ellis and Dave Marett present enhancements to the Archimedes style desktop utility.

The ADFS desktop utility (BEEBUG Vol.8 No.10) is an excellent 'front-end' and has prompted us to transfer all our programs from the DFS to the ADFS system because of its ease of use.



Revised desktop display

Having used the program for a while we decided that it would be a more complete utility if some extra facilities were added. Experimentation with the program has led us to produce the enhancements now described.

Briefly these are:

- Create Directory
- Select Drive
- Rename Directory/File
- Delete Directory/File
- Display Time

Also displayed are the day/date and the bytes free on the currently selected disc. In order to allow for these enhancements the hex/ASCII dump routine is lost, as is the Quit command.

The listing given here shows the changes to be made to the previous version of the program *Desk2*. To implement the changes, load in the previous version of

Desk2, delete lines 1520 to 1550 inclusive using **DELETE 1520,1550** and then type in the listing shown here. Make sure you keep exactly to the line numbers as given, because some lines will replace lines from the previous version, and some lines are new. When you have finished, save the complete program as *Desk2*. It is a good idea to rename your original program, so that if you make any errors in entering the program, the original has not been overwritten.

The program works in exactly the same way as the original one so that, for example, before you can delete or rename a file, it must first be 'clicked' and highlighted.

We have found that the program enhancements add to an already useful 'front-end' utility, and feel it is well worth the hour or so spent in updating the original *Desk2*.

PROGRAM DETAILS

Deleting lines 1520 through 1550 remove the hex/ASCII dump routine.

Lines altered are:

- 100 - New Error and message procedure
- 120 - Gives new top Command line
- 1040 - Inserts new variables D, bytes\$, ren1\$, del1\$, cdir\$
- 1060 - Adjusted to take into account new line 1065
- 1130 - Y% test reduced by 2 so that the pointer does not go down as far as the new information line
- 1430 - Print Tabs now inserted to complement new procedures
- 1630 - Print Tabs inserted and OSCLI command enhanced

ADFS Desktop Enhancements

New lines are:

- 70 - To ensure the Root Directory is accessed first
- 1065 - New bottom information line
- 1195 - Accesses the new procedure to give 'Bytes Free'
- 1325 - The title bar
- 1454 } Revamped PROCOptions to allow
- 1458 } for new commands and altered VPOS variables

New procedures are:

PROCbytes - Calculates the number of bytes free on the currently selected disc.

PROCdrive - Selects either Drive 0 or Drive 1.

PROCmessage - Basically an error trap routine, e.g. when trying to delete a non-empty directory.

PROctime(x) - Gives a time delay on some messages.

PROCclk - Activates the digital clock on the Command Line, and then resets, after a short time delay.

PROctitleboxa and **PROctitleboxb** - Draw the new title box with its shadow box.

PROCcdir - Allows for the creation of new directories.

PROCrename - Renames files and directories.

PROCdelete - Deletes named files and directories.

The updated version of the ADFS desktop is available on this month's magazine disc. This consists of two programs: *Desktop*, which is unchanged from before, and the enhanced *Desk2* program.

```
20 REM Version B1.1
40 REM BEEBUG October 1991
55 REM Upgrade Terry Ellis & Dave Mar
ett
```

```
70 *DIR $
100 ON ERROR PROCmessage
120 PROCtext (" [ ] Show"+STRING$(4, " "
)+ "Info"+STRING$(4, " ")+"Cdir"+STRING$(4
, " ")+"Change Disc"+STRING$(3, " ")+"Driv
e"+STRING$(3, " ")+"Rename"+STRING$(3, " "
)+ "Delete"+STRING$(3, " ")+"Time", 20, 1020
)
1040 DEFPROCinit:dx1%=1:dx2%=38:dy1%=28
:dy2%=2:z%=255:scr%=0:bytes$="" :DIMf$(46
),t$(46):ren1$="" :del$="" :cdir$="" :D=0:s
lot%=4:ENDPROC
1060 DEFPROCclear:X%=1:Y%=0:PROCread:VD
U17,128,17,0,12,17,128,17,1,88:GCOL0,1:M
OVEx1%,y2%-32:DRAW x2%,y2%-32:MOVEx1%,y2
%-830:DRAW x2%,y2%-830
1065 PROCtext ("Date - "+LEFT$(TIME$,15)
+STRING$(7, " ")+"Bytes Free - "+bytes$+S
TRING$(7, " ")+"Drive Selected - "+STR$(D
),50,130):IF T$<>" PROCstring(T$,2,0)
1130 IFINKEY-42 IF Y%<27 CALL&903:Y%=Y%
+1:CALL&900
1195 PROCbytes
1325 PROctitleboxa:PROctitleboxb:GCOL0,
1:PROCtext (" A D
F S - D E S K T O P
",40,55)
1430 DEFPROCromload:CLS:OSCLI"FX15":PRI
NTTAB(6,10)"Load ROM image into which"
1432 INPUTTAB(9,12)"sideways RAM slot "
;slot%:IFslot%<4 OR slot%>7 PRINTTAB(28,
12) " ":GOTO1432
1434 OSCLI"SRLOAD "+f$(S%)+ " 8000 "+STR
$slot%:PROCclear:ENDPROC
1450 DEFPROCOptions:x%=X%-2
1454 IFx%DIV4=0 ANDs%<255 PROCshow ELSE
IFx%DIV4=1PROCinfo ELSEIFx%DIV4=2 PROCcd
ir ELSEIFx%DIV4=3 ORx%DIV4=4 PROCnew
1458 IFx%DIV4=5 PROCdrive ELSEIFx%DIV4=
6 PROCrename ELSEIFx%DIV4=7 PROCdelete E
LSEIFx%DIV4=8 PROCclk
1570 DEFPROCkey:OSCLI"FX15":PRINTTAB(12
,23)"Press any key";:Z%=GET:PROCclear:CO
LOUR129:COLOUR0:PROChighlt(S%):COLOUR128
:COLOUR1:ENDPROC
2000 :
2010 REM ----- NEW PROCEDURES -----
2020 :
2030 DEFPROCbytes
2040 bytes=0:bytes$=""
```

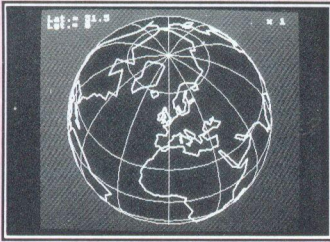

ADFS Desktop Enhancements

```
2050 A%=&71:X%=&80:Y%=0
2060 CALL &FFF1
2070 bytes=!&80
2080 bytes$=STR$(bytes)
2090 ENDPROC
2100 :
2110 DEFPROCdrive
2120 IF D=0 D=1 ELSE D=0
2130 CLS:PRINTTAB(13,10)"Drive Number"TAB
AB(18,12);D TAB(15,14)"Selected":PROCTim
e(50)
2140 PROCnew:ENDPROC
2150 :
2160 DEFPROCmessage
2170 CLS:PRINTTAB(9,10)"Command not Acc
epted":PROCTime(100)
2180 RUN
2190 ENDPROC
2200 :
2210 DEFPROCtime(x)
2220 TIME=0:REPEAT UNTIL TIME=x:ENDPROC
2230 :
2240 DEFPROCclk
2250 GCOL0,0:PROCText("Time",1125,1020)
:GCOL0,1:PROCText(MID$(TIME$,17,5),1125,
1020)
2260 PROCTime(150)
2270 GCOL0,0:PROCText(MID$(TIME$,17,5),
1125,1020):GCOL0,1:PROCText("Time",1125,
1020)
2280 ENDPROC
2290 :
2300 DEFPROCTitleboxa
2310 GCOL0,1:MOVE19,10:DRAW19,49:PLOT85
,1235,49:MOVE1235,49:DRAW1235,10:PLOT85,
19,10
2320 ENDPROC
2330 DEFPROCTitleboxb
2340 GCOL0,0:MOVE35,20:DRAW35,59:PLOT85
,1248,59:MOVE1248,59:DRAW1248,20:PLOT85,
35,20
2350 ENDPROC
2360 :
2370 DEFPROCcdir
2380 CLS:OSCLI"FX15":PRINTTAB(10,10)"Co
nfirm Create a"TAB(9,12)"New Directory Y
/N"
2390 G$=GET$:IF G$="Y" OR G$="y"GOTO 24
00 ELSE PROCkey:ENDPROC
2400 CLS:PRINTTAB(9,10)"Enter New Direc
```

```
tory"TAB(9,12)"Name (max 10 chrs)"TAB(1
2,15)"> <"
2410 INPUTTAB(13,15)cdir$
2420 OSCLI"CDIR "+cdir$
2430 CLS:PRINTTAB(12,10)"New Directory"
TAB(13,12)cdir$ TAB(14,14)"Created : "
2440 cdir$="" :PROCTime(100):PROCkey
2450 ENDPROC
2460 :
2470 DEFPROCrename
2480 IF S%=255 ENDPROC
2490 IF t$(S%)=0 CLS:OSCLI"FX15":PRINTT
AB(7,10)"Confirm Rename Directory "TAB(1
4,12)f$(S%)TAB(16,14)"(Y/N)"
2500 IF t$(S%)>0 CLS:OSCLI"FX15":PRINTT
AB(10,10)"Confirm Rename File"TAB(14,12)
f$(S%)TAB(16,14)"(Y/N)"
2510 G$=GET$:IF G$="Y" OR G$="y" GOTO25
20 ELSE PROCkey:ENDPROC
2520 CLS:PRINTTAB(8,10)"Enter Replaceme
nt Name"TAB(13,12)"(max 10 chrs)"TAB(13,
14)"> <"
2530 INPUTTAB(14,14)ren1$
2540 OSCLI"ACCESS "+f$(S%)+ " WR"
2550 OSCLI"RENAME "+f$(S%)+ " "+ren1$
2560 OSCLI"ACCESS "+ren1$+" LWR"
2570 CLS:PRINTTAB(10,10)"File or Direct
ory"TAB(13,12)f$(S%) TAB(10,14)"<<<< Ren
amed >>>>"TAB(13,16)ren1$
2580 ren1$="" :ren2$="" :f$(S%)="" :S%=255
:PROCTime(100):PROCkey
2590 ENDPROC
2600 :
2610 DEFPROCdelete
2620 IF S%=255 ENDPROC
2630 IF t$(S%)=0 CLS:OSCLI"FX15":PRINTT
AB(7,10)"Confirm Delete Directory"TAB(14
,12)f$(S%)TAB(16,14)"(Y/N)"
2640 IF t$(S%)>0 CLS:OSCLI"FX15":PRINTT
AB(10,10)"Confirm Delete File"TAB(14,12)
f$(S%)TAB(16,14)"(Y/N)"
2650 G$=GET$:IF G$="Y" OR G$="y" GOTO 2
660 ELSE PROCkey:ENDPROC
2660 OSCLI"ACCESS "+f$(S%)+ " WR"
2670 OSCLI"DELETE "+f$(S%)
2680 CLS:PRINTTAB(14,10)del$ TAB(15,12)
"Deleted"
2690 del$="" :f$(S%)="" :S%=255:PROCTime(
100):PROCkey
2700 ENDPROC
```

B

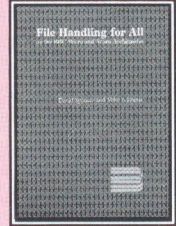
Applications I Disc



- 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.

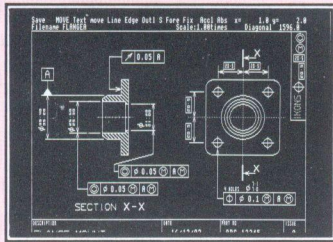
The topics covered by the book include:

- 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.

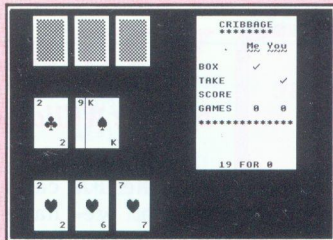
ELEVENENSE - another popular version of Patience - lay down cards on the table in three by three grid and start turning them over until they add up to eleven.

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.

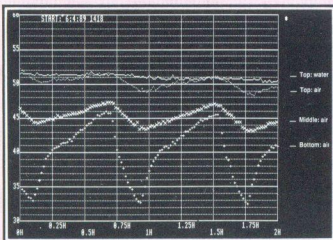
TWIDDLE - a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern.

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 II 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.

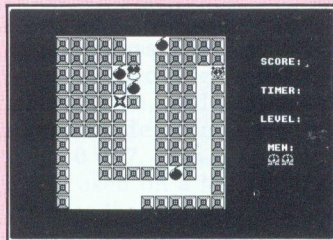
PITFALL PETE - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way.

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		Stock Code	Price
Arcade Games (40/80 track DFS)	PAG1a	£ 5.95	Arcade Games (3.5" ADFS)	PAG2a	£ 5.95
Board Games (40/80 track DFS)	PBG1a	£ 5.95	Board Games (3.5" ADFS)	PBG2a	£ 5.95
File Handling for All Book	BK02b	£ 9.95			
File Handling for All Disc (40/80T DFS)	BK05a	£ 4.75	File Handling for All Disc (3.5" ADFS)	BK07a	£ 4.75
Joint Offer book and disc (40/80T DFS)	BK04b	£ 11.95	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



VDU and FX Calls (6)

by Mike Williams

All the FX calls which we have looked at so far have performed some function for the user. You simply issue the appropriate FX call and that's an end to it. However, some FX calls (or their equivalent OSBYTE calls) are used to *obtain* information. They therefore differ from previous calls in that there has to be some mechanism for returning this information to the program from which the call has been made.

This requires some different techniques, and we can perhaps deal with this best by means of an example. Basic provides two functions called POS and VPOS which return the current position of the text cursor. This can be quite useful if you need to move from the current position of the cursor in order to perform some function, and then return to the original position. Simply save the values supplied by POS and VPOS before the move, and then use these to reposition the cursor later:

```
X=POS: Y=VPOS
PROCdo_task
VDU31,X,Y
```

There is also another function, POINT, which can be used in graphics work which will tell you the colour of the pixel at the current position. But when dealing with text, there is no function in Basic which will tell you the character at any particular position on the screen. However, there is an FX (OSBYTE) call which will do this for you.

The FX function itself is only designed to call an operating system routine, not to return any information. So to achieve this we have to take a different approach.

Basic provides two keywords which come to our aid, CALL and USR.

CALL is in some ways a bit like FX - it is used to call a routine in machine code. This could be an operating system routine, or any other including those written by yourself. All FX calls use the same basic routine, called OSBYTE, at address &FFF4 (a hexadecimal address). A particular FX call is invoked by assigning its number to A% (representing the accumulator) before calling the OSBYTE routine. In addition, any other parameters are first assigned to X% and Y% (representing the processor's X and Y registers). Thus the call:

```
FX4,1
```

could also be written as:

```
A%=4
```

```
X%=1
```

```
CALL(&FFF4)
```

Clearly, the FX format is simpler when programming in Basic.

The USR function on the other hand is used when information is going to be returned, usually in the format:

```
Z%=USR(&FFF4)
```

Z% is a four byte integer, and it is possible for each byte to contain a different piece of information.

To read a character at a given position, we need to move to that position, and then call OSBYTE having first set A%=135, thus:

```
VDU31,X,Y
```

```
A%=135
```

```
Z%=USR(&FFF4)
```

```
char=(Z% AND &FFFF) DIV &100
```

At the conclusion of this piece of code, the variable *char* will contain the ASCII value of the character in position (X,Y) on the

screen. If there is no character at that position, or it cannot be recognised as a valid character, the value returned will be zero. Let's look at this in more detail so that we can understand what is going on. The first line is a VDU call which moves the text cursor to the position (X,Y) on the screen at which we want to read a character. A% is set to 135, as it is in effect FX135 which we need to use. The third line calls the OSBYTE routine so that the information we want is stored in Z%. Then comes the tricky part.

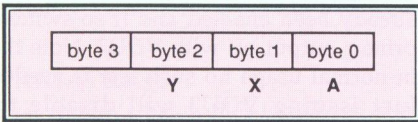


Figure 1. Format of data returned by the OSBYTE function

We have seen that four bytes of information are returned, including the values of the A, X, and Y registers (see figure 1). OSBYTE 135 returns in the X register the ASCII value of the character found, and we thus have to isolate this from the other three bytes. This is the purpose of the last line. ANDing with &FFFF leaves us with the bottom two bytes only (by masking out the other bits), and dividing by &100 moves all these bits eight places to the right, so that the value of A disappears and we are left only with the value of the X register.

Now all of this may seem hopelessly complicated. I have tried to explain how it all works, but to use it you don't have to understand it at all. Just incorporate the above four lines in your program whenever you need to read a character from the screen. In fact, we can do even better than this by packaging the whole routine up as a function:

```
DEF FNchar (X,Y)
LOCAL A%,X1,Y1,Z%
```

```
X1=POS:Y1=VPOS
VDU31,X,Y
A%=135
Z%=USR(&FFF4)
VDU31,X1,Y1
=(Z% AND &FFFF) DIV &100
```

This function can be used to return the ASCII code of the character in any position (X,Y) on the screen, and has the added refinement that by using POS and VPOS the routine 'remembers' where the text cursor was to start with, and ensures that it is returned to that position on exit. If you want the character itself, rather than its ASCII code, then use the CHR\$ function to convert from one to the other.

Although all this may seem somewhat complicated compared with the simpler form of FX call, remember that there is simply no other way of performing this particular function, and packaged up as a function makes life as easy as possible.

READING SCREEN MODE

As it happens, OSBYTE 135 can also be used to provide another piece of information, and that is the current screen mode in use (a number in the range 0 to 7, even if a shadow mode is being used on a Master or Compact). Whereas the character at the cursor was returned in the X register, the screen mode is returned in the Y register, and this is the third byte from the right of the four bytes returned.

We can therefore write another function, very similar to that above, to provide this new information:

```
DEF FNmode
LOCAL A%,Z%
A%=135
Z%=USR(&FFF4)
=(Z% AND &FFFFFF) DIV &10000
```

First Course

This could be useful in any situation where the detail of your program is mode dependent, for example, setting a text window. To distinguish between 40 and 80 column modes you could write:

```
mode=FNmode
IF mode=0 THEN
    VDU28,0,31,79,20:
    LL=80
ELSE
    IF mode=1 OR mode=4 THEN
        VDU28,0,31,39,20:
        LL=40
```

This has been laid out to make the meaning as clear as possible - in practice the coding would form one more or less continuous line of Basic. For mode 0 the coding sets a text window 80 characters wide and sets the variable *LL* to 80 for future use; if the screen mode is 1 or 4, then the text window is appropriately 40 characters wide, and *LL* is set to 40.

VDU STATUS

Another example of this approach to the use of FX calls is OSBYTE 117 (available only on the Master and Compact systems) which enables a program to find the value of the so-called VDU status byte. Each of the eight bits making up this byte can tell you something about the current state of your system as listed in figure 2.

The status byte is returned in the X register as with the previous example, but this time we will also need to isolate a particular bit to check. Here is a function which will do this:

```
DEF FNvdu_byte(b%)
LOCAL A%,Z%
A%=117
Z%=USR(&FFF4)
```

```
Z%=(Z% AND &FFFF) DIV &100
=(Z% AND 2^b%)=1
```

The parameter *b%* indicates which bit is to be tested (in the range 0 to 7), and the function as written (see the last line) returns a value of TRUE or FALSE depending on whether the corresponding bit is set or not. Thus we could write:

```
IF FNvdu_byte(0) THEN VDU3
```

Which checks to see if the printer has already been enabled and if so switches printer output off with VDU3. Note that in normal usage no such test is needed. Just issuing VDU3 will disable the printer if enabled, and otherwise have no affect. However, the example does show how the function could be used.

Bit	Status when set (i.e. bit set to 1)
0	Printer enabled by VDU2
1	Scrolling disabled
2	Paged scrolling selected
3	Text window in force
4	Currently in shadow mode
5	Printing at graphics cursor enabled with VDU5
6	Cursor editing in progress
7	Screen disabled with VDU21

Figure 2. Contents of VDU status byte

That concludes our discussion of VDU and FX calls which has run to five separate articles, and there are still many more FX calls (or equivalent OSBYTE calls) which we haven't covered. As I said last month, many if not most of these will never be needed, but I hope the examples which have been discussed in this series will have given you the confidence to explore other FX calls should be need arise, and use the framework of the examples in these articles as the basis for using such other calls in your own programs. **B**



512 Forum

by Robin Burton

This month let's tidy up a few 'left-overs' from the last two issues. In spite of the fact that each of those Forums was slightly extended some relevant points weren't covered.

You might have been surprised by what can be done in a short batch file, but this month let's consider what can't be done.

VARIATIONS

Last month's batch files employed some of the less common functions, but even so others remained unused, like 'FOR <variable> IN (set) DO <command>'. This is probably less understood than the functions we've already looked at.

Here's another batch file, similar to the second one last month, but in spite of the much more complex line 3 it does much the same job, though in a subtly different way.

```
COPY %0.BAT %1
:START FOR %%F IN
(???????) DO COPY %%F %1%%F
IF %1 == D EXIT
COPY %1 %2
SHIFT
GOTO START
```

Just as before the job is called with a number of parameters, so you can try it out easily by referring to last month's article. As we saw then, parameters are accessed by means of numbers zero to nine preceded by a '%' symbol, while the contents of parameter variables can be changed by the 'SHIFT' command.

General variables (such as %%F above) are specified by two percentage symbols and any single alphanumeric character. Because the contents of these variables are entirely under the control of your commands, rather than being fixed at the stage of command entry, in this case DOS doesn't care what you call them or the order in which they're defined and used. However, the penalty for this freedom is that the only way to set up the contents of general variables is by means of the 'FOR <variable> IN (SET)' construct, which translates roughly to: 'FOR (each value of <variable>) IN (the) (SET)' DO something...

Paraphrasing line 3, it reads "FOR each occurrence of the name '%%F' derived from the set of files defined by (???????) DO the following... COPY the file named in variable %%F to a new file which is to be called %1%%F."

The names indicated by (SET), which by the way is always enclosed in brackets, must evaluate to filenames. These can be specified with or without an extension (or vice-versa for that matter) and any part of the filename and/or extension in (SET) can be hard coded as literals, can include parameters as well as wildcards, or can be any meaningful combination of them.

For example (?%1?%2?ABC.%3Y) could, in the right circumstances, represent a valid (SET). Certainly the expression is valid syntactically so long as the contents of parameters %1 and %2 don't exceed a single character and %3 doesn't exceed two characters in length.

On the face of it then, it looks like 'FOR IN (SET) DO' could be quite a powerful facility, especially when you remember tests like '==', 'SHIFT' (and 'IF EXIST' which we haven't examined recently).

You'll find however, if you try the above batch file, that in practical terms the results are identical to our simpler version last month, where the equivalent to line 3 was:

```
COPY ??????? %1*
```

The major difference in this version of the file is that when the third line executes, the 'COPY' command plus the source and destination filenames are interpreted and displayed in full for every file copied. A by-product of this is that the operation will also be slightly slower, though if you did run last month's job you'll know that this point is somewhat academic in this context.

The reason for the difference in operation is fairly subtle and an explanation might not help much, but let's try. It hinges on the fact that the current value of %%F is always fully evaluated before it is used in each command. At all times therefore, %%F represents an explicit value which can refer only to a single file. In other words, the evaluated contents of general variables do not and cannot include wildcards, although the (SET) in which they are defined might very well do so, as we see here.

In consequence, at run time the 'DO' part of the command is issued repeatedly for each interpreted value of %%F, that is, for every file copied, with full and unique source and destination filename supplied every time. One file is copied and that

COPY command ends immediately, although it may be re-issued at once by the first part of the line if further evaluation of %%F requires it. In our example here it usually does, and the reason the job is slower is because each fresh COPY command has to locate the (next) source file in the directory all over again.

By contrast, in last month's batch file, the source names in the command consisted literally of wildcards, so only one command was needed for each set of files. Each copy command therefore executed until all qualifying source files had been processed. Because all the files in each set were copied in one command, DOS kept a pointer into the directory of the last file processed, so no search of the directory was needed between the last file copied and the next. Only one search was required for the first file in each set and execution was therefore quicker (I told you it was subtle).

LIMITATIONS

In spite of the complication of 'FOR IN (SET) DO', in my opinion it's difficult to think of realistic situations which require it which can't be catered for just as easily using simpler commands and wildcards (if you can think of any I'd be interested to hear them).

As we saw before, you can test the contents of parameters against hard coded values (or other parameters or a combination), and make decisions based on equality or otherwise. You might hope therefore, that general variables would allow access to the (SET) name last processed, but if so you'd be disappointed.

The only way to 'define' the contents of general variables is the method used above. What's more you can't use them outside the line which defines them. In effect the interpreted variable contents exist only while each 'DO' command is being executed. You can't therefore, as an example, replace line four in the job above by:

```
IF %%F == DDDDDDD EXIT
```

It would be useful if this would work, giving the same result as our original test of 'IF %1 == D', but if you try it you'll see it simply doesn't work.

I always think that several batch functions seem distinctly 'unfinished' and that 'FOR IN (SET) DO' is merely an answer looking for a question. This is a pity, because with a bit more effort batch files could have been much more useful. Instead the problems, or more accurately the restrictions of DOS's batch language are, in the event, far too rigid for it to be used as a general purpose vehicle.

Consider! Parameters can be SHIFTed left, but every time you do this you lose the existing %0 value and you can neither save it by any means nor recall it later, since there's no UNSHIFT function. Equally, sometimes it would be useful to be able to say something like:

```
%%A = %%B or %%A = %1 or even %1 = "A BC"
```

This looks simple enough, but it can't be done. There are lots of other shortcomings in the batch language which seem glaringly obvious when compared with even the simplest of interpreted languages. You can't nest

'FOR <variable> IN (SET) DO' commands, only one at a time can be active. The (single) command in the 'DO' portion must be contained entirely in the current line, and it may not be conditional. Multiple commands (e.g. several 'DO' lines) therefore can't be used because there's no 'NEXT' to tell each 'FOR' where to end, while 'IF' can't follow 'DO' because there's no 'ELSE'.

PITFALLS

To be fair, by stretching your imagination a bit you can construct moderately complex operations in batch files. However, you must take extreme care that the files actually do what you expect them to if you're trying to get clever. Complex batch files can be unreliable. You may think I'm being harsh, but try this. Create a few dummy files called FILE1.TST, FILE2.TST and so on. It doesn't matter what they contain and just two or three files will be enough. Then create and run the following batch file:

```
FOR %%A IN (FILE?.TST) DO COPY %%A %% ABCD
```

By the way, you could alternatively take a short cut and issue this as an immediate, manual command. To use general variables in manual commands you should specify them with only one '%' symbol in each variable, like this:

```
FOR %A IN (FILE?.TST) DO COPY %A %ABCD
```

I know this command looks a bit silly, but its purpose is to highlight the command processor's limited intelligence. You can see at once that the 'BCD' on the end of the line is a mistake, but the point is that the command

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

Edikit (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)

processor can't. It 'thinks' the extra characters make the destination filename different to the source name. They don't of course, because the 'BCD' is completely spurious and has no effect on the names, but try the command and you'll see that it happily copies each of the files in the list to itself!

The destination names will of course include the 'BCD' in the screen display, but this is truncated when the filename is stored on the disc. The assumption (wrong in this case) is that if a command is evaluated and validated by the command processor with no error it must be both sensible and correct.

This command clearly demonstrates that this needn't be true. That's why I warned you to be careful if you get adventurous with batch files, especially using parameters, when the contents of run-time variables may not be obvious. A 'DIR' after the test above will demonstrate that you've no more files than you started with, although every file has been copied.

OK, so this example is harmless and somewhat artificial. However, think of the potential dangers if you expected a batch file to copy real files to new ones, but in the event it didn't? This could be a bit more serious don't you think?

Weighing up the capabilities versus the limitations of batch functions I keep coming back to the thought that with modest extra effort the designers could easily have produced a much more powerful, useful and above all reliable facility. The basic ideas seem fine, but the job looks half finished, as if someone went to lunch and forgot to return to complete it.

This glimpse of what might have been needn't be a permanent frustration, but you'll have to go to outside software suppliers to find a cure. There are several batch file utilities and compilers for DOS systems, all of which were produced because of these shortcomings, bugs and omissions in the standard system.

If you've ever felt frustrated by the limitations of batch files and want to enhance them beyond the facilities offered by the standard implementation you'll find various worthwhile batch file utilities and compilers in shareware at the usual very reasonable prices.

CONTACTS

Finally, I have details of a couple more members who would like to hear from other 512 users and have asked for their addresses to be published.

First is:

Mr. C.W. Robertson,
Three Gables, 7A Talbots Drive,
Maidenhead, Berks SL6 4LZ.

The second is Oliver Debus. Oliver is 19 years old and would like to hear from other 512 users, most especially of course if they also live in Germany, but Oliver's English is excellent so don't let that stop you writing. He tells me the nearest Acorn dealer is 100Km away and deals only in Archimedes systems anyway, so Oliver feels just as cut off as Ron Thompson who I mentioned a few months back. Write to Oliver at:

Muehlberg 18, D-8871 Harthausen,
Federal Republic of Germany.

B

Double Size Characters in Mode 0

David Stevens describes two routines for creating double size characters in 80 column modes.

Double height characters can look a bit cramped in mode 0. A more pleasing effect can be achieved by creating double sized (double-height double-width) characters.

The Basic procedure (PROCdbslz) given in listing 1 does the job, but it is a bit slow. The much faster assembler version (PROCdbl) in listing 2 uses more memory, and so may be of more use in mode 128 on the Master. If you use the assembler version make sure that PROCdcode is called before PROCdbl to assemble the code. Both listings present the relevant procedures in complete demo programs.

Each of the two procedures has the same 3 parameters: the text string and the X and Y text co-ordinates of the top left position of the first character. As listed, both versions use characters 200 to 203 for the four quarters of the expanded characters. If you wish to use other characters simply change the numbers in lines 1100 to 1140 of listing 1, or lines 1250 to 1300, and 1340 of listing 2.

The usual warning applies to listing 2 - because of the assembler save it before you attempt to run it.

Listing 1

```
10 REM Program Dblsiz1
20 REM Version B1.0 (Basic)
30 REM Author David Stevens
40 REM BEEBUG October 1991
50 REM Program subject to copyright
60 :
100 MODE0
110 DIM L% 8,R% 8
120 T$="DOUBLE SIZE"
130 PROCdbslz(T$,40-LENT$,3):PRINT''
140 END
150 :
1000 DEF PROCdbslz(A$,x%,y%)
```

```
1010 LOCAL A%,X%,Y%,j%,k%,n%
1020 A%=10:X%=R%MOD&100:Y%=R%DIV&100
1030 VDU31,x%,y%
1040 FOR j%=1 TO LENA$
1050 ?R%=ASC(MID$(A$,j%,1)):CALL&FFF1
1060 FOR k%=1 TO 8
1070 n%=(R%?k%)DIV&10:L%?k%=(n%AND8)*24
+(n%AND4)*12+(n%AND2)*6+(n%AND1)*3
1080 n%=(R%?k%)MOD&10:R%?k%=(n%AND8)*24
+(n%AND4)*12+(n%AND2)*6+(n%AND1)*3
1090 NEXT k%
1100 VDU23,200,L%?1,L%!1;L%!2;L%!3;L%!4
1110 VDU23,201,R%?1,R%!1;R%!2;R%!3;R%!4
1120 VDU23,202,L%?5,L%!5;L%!6;L%!7;L%!8
1130 VDU23,203,R%?5,R%!5;R%!6;R%!7;R%!8
1140 VDU200,201,10,8,8,202,203,11
1150 NEXT j%
1160 ENDPROC
```

Listing 2

```
10 REM Program Dblsiz2
20 REM Version B1.0 (Assembler)
30 REM Author David Stevens
40 REM BEEBUG October 1991
50 REM Program subject to copyright
60 :
100 MODE0
110 PROCdcode
120 T$="DOUBLE SIZE"
130 PROCdbl(T$,40-LENT$,3):PRINT''
140 END
150 :
1000 DEF PROCdbl($text%,?xd%,?yd%)
1010 CALL big%
1020 ENDPROC
1030 :
1040 DEF PROCdcode
1050 DIM big% 450
1060 xd%=&80:yd%=&81:char%=&82:ptr%=&83
1070 num%=&84:store%=&85:temp%=&86
1080 mult%=&87:prod%=&88
1090 FOR pass=0 TO 2 STEP 2
1100 P%=big%
1110 [OPT pass
1120 :
1130 \ position cursor
```


Double Size Characters in Mode 0

```
1140 :
1150 .big% LDA #31:JSR &FFEE
1160 LDA xd%:JSR &FFEE
1170 LDA yd%:JSR &FFEE
1180 :
1190 \ create skeleton VDU sequence
1200 :
1210 LDX #0:LDA #23:STA vdu%,X
1220 LDX #10:STA vdu%,X
1230 LDX #20:STA vdu%,X
1240 LDX #30:STA vdu%,X
1250 LDX #1:LDA #200:STA vdu%,X
1260 LDX #11:INA:STA vdu%,X
1270 LDX #21:INA:STA vdu%,X
1280 LDX #31:INA:STA vdu%,X
1290 LDX #40:LDA #200:STA vdu%,X
1300 INX:INA:STA vdu%,X
1310 INX:LDA #10:STA vdu%,X
1320 INX:LDA #8:STA vdu%,X
1330 INX:STA vdu%,X
1340 INX:LDA #202:STA vdu%,X
1350 INX:INA:STA vdu%,X
1360 INX:LDA #11:STA vdu%,X
1370 :
1380 \ read each character
1390 :
1400 STZ char%
1410 .nextchar%
1420 LDX char%:LDA text%,X
1430 CMP #&0D:BNE process%
1440 RTS
1450 :
1460 \ get its definition via osword
1470 :
1480 .process%:STA block%
1490 LDA #10:LDX #block% MOD&100
1500 LDY #block%DIV&100:JSR &FFF1
1510 :
1520 \ divide into 4 characters
1530 :
1540 STZ ptr%:LDY #1
1550 .looptl% INC ptr%:INY:LDX ptr%
1560 LDA block%,X:PHA:TYA:TAX:PLA
1570 LSR A:LSR A:LSR A:LSR A
1580 JSR expand%
1590 LDA ptr%:CMP #4:BNE looptl%
1600 STZ ptr%:LDY #11
1610 .looptr% INC ptr%:INY:LDX ptr%
1620 LDA block%,X:PHA:TYA:TAX:PLA
1630 AND #&F:JSR expand%
1640 LDA ptr%:CMP #4:BNE looptr%
```

```
1650 LDA #4: STA ptr%:LDY #21
1660 .loopbl% INC ptr%:INY:LDX ptr%
1670 LDA block%,X:PHA:TYA:TAX:PLA
1680 LSR A:LSR A:LSR A:LSR A:JSR expand
%
1690 LDA ptr%:CMP #8:BNE loopbl%
1700 LDA #4:STA ptr%:LDY #31
1710 .loopbr% INC ptr%:INY:LDX ptr%
1720 LDA block%,X:PHA:TYA:TAX:PLA
1730 AND #&F:JSR expand%
1740 LDA ptr%:CMP #8:BNE loopbr%
1750 :
1760 \ output VDU sequence
1770 :
1780 LDX #0
1790 .vduloop% LDA vdu%,X:JSR &FFEE
1800 INX:CPX #48:BNE vduloop%
1810 INC char%:JMP nextchar%
1820 :
1830 \ space for string, VDU sequence
and osword parameter block
1840 .text%:] :FORj%=P%TOP%+40:?j%=0:NEX
T:P%=P%+40:[OPT pass
1850 .vdu%:] :FORj%=P%TOP%+48:?j%=0:NEXT
:P%=P%+48:[OPT pass
1860 .block% EQU0 0:EQU0 0:EQU0 0
1870 :
1880 \ integer multiplication
1890 :
1900 .multiply%
1910 PHX:LDX mult%:DEX:LDA temp%
1920 .next% CLC:ADC temp%:STA prod%
1930 DEX:BNE next%
1940 PLX:CLC:LDA store%
1950 ADC prod%:STA store%:RTS
1960 :
1970 \ expand each quarter of character
and insert in VDU squence
1980 :
1990 .expand% STA num%
2000 AND #8:STA temp%:STZ store%
2010 LDA #24:STA mult%:JSR multiply%
2020 LDA num%:AND #4:STA temp%
2030 LDA #12:STA mult%:JSR multiply%
2040 LDA num%:AND #2:STA temp%
2050 LDA #6:STA mult%:JSR multiply%
2060 LDA num%:AND #1:STA temp%
2070 LDA #3:STA mult%:JSR multiply%
2080 STA vdu%,Y:INY:STA vdu%,Y:RTS
2090 ]
2100 NEXT:ENDPROC
```

B

BEEBUG Education

This month, Mark Sealey reviews Selladore Tales from Sherston Software.

Product	Selladore Tales
Supplier	Sherston Software Swan Barton, Sherston, Malmesbury, Wiltshire SN16 0LH. Tel. (0666) 840433
Price	£24.00 ex. VAT

Selladore Tales comes from education specialists Sherston Software, and is in the best tradition established for this type of product. There are versions for the A3000/Archimedes (which cost the same) as well as 8-bit machines. It is a review of the latter, of course, that forms the substance of this month's Beebug Education.

The package comes on two discs. The start-up disc is copyright protected and cannot therefore be copied. The 'user' disc should be copied. Both are 40 track DFS format. Also in the A5 size plastic wallet are a slim teachers book, a dozen or so activity and word sheets, a keystrip and an attractively produced version of the text.

Selladore Tales is described as "an easy read adventure" by the package's authors. It is designed for use by younger children with reading difficulties - especially those at Upper Junior level.

This is reflected in the quality of the contents - the subject matter will appeal to children of 10 or 11 years, and older. The language content and accompanying work, though, is more appropriate to those with a slightly lower reading age. There is also a big-print version for pupils who may prefer it or who are partially sighted. Back and foreground colours are selectable for extra clarity. It is possible to use a Concept Keyboard too.

The story has two components. The "Curse of Zorin" is a text adventure which introduces the main characters and leaves them in a predicament taken up as the computer game, "Black River Quest". The genre is not unlike Lord of the Rings, full of action and fairy-tale plot, but with a storyline that neither patronises nor bores.

USING THE DISCS

Booting the start-up disc happens in the normal way; if you are using a machine with sideways RAM, you can take advantage of this. The teacher's book also explains that certain ROMs can clash with *Selladore Tales* and explains how to overcome this. After a few seconds (during which there is some measure of feedback on the booting progress) you are prompted to remove the start-up disc and insert the user disc.

It is possible at this stage to reload an old, previously saved, game. Otherwise you start from scratch. The screen now contains text and graphics. The graphic at the top gives a clue as to your location and the text lower down contains three elements: the last instruction you issued, messages from the adventure itself and a prompt for your next instruction.

THE ADVENTURE

In many respects the program behaves like the conventional "take lamp"/ "go south" text adventure. Upper and lower case are both accepted. A card lists the keywords to which the program will respond and the usual "list" and "stop" commands.

The graphics are imaginative and plain, carry meaning but take between about 3 and 5 seconds to load - every time. This could become a slight annoyance when all the pupil(s) are doing is routing back

over familiar territory. In other respects, the screen is clear and appealing. The keyboard buffer is flushed at each entry so as to prevent errors due to enthusiastic typers ahead.

```

You are in a damp cave.
You can see
an old beggar.
You can go south.
-----
You wanted to: go north
This man is very poor.
What now?
  
```

Selladore Tales text dialogue

On the whole, the adventure is challenging and will involve most children, though there is at least one inconsistency between the spellings of written cues (in the teacher's book) and what is accepted on screen ('mellon', 'melon').

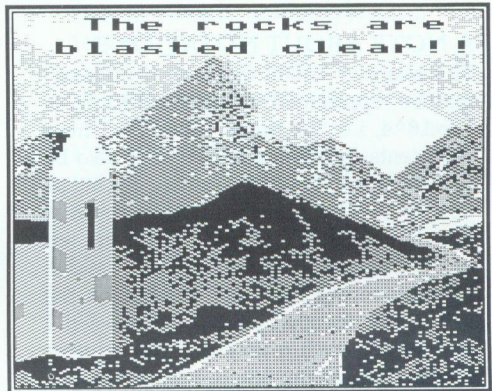
MAKING MORE OF THE SOFTWARE

As is the case with much educational software of this kind, it is what goes on away from the computer - or how another use altogether is made of it - that helps to single out the good from the bad. A text file of the story is available and can be used in the formats of various other programs. The manual explains quite clearly how to copy these onto a second blank disc before using them with Pendown, Writer, Folio, Interword and WordWise Plus.

More significantly still, the text can be loaded into that excellent suite Developing Tray. This will allow various types of interactive language activity to be performed on it, including cloze (basically

gap filling) and quasi cloze procedure. It will also allow material with which children will be relatively familiar (having used and read Selladore Tales) to be investigated with all the near magic of what Tray can do.

In the classroom, use will depend on teaching style, resources and the children and their needs. Selladore Tales could form the backbone to a topic on fairy lands, dragons and 'the unknown'. It would be helpful to allow other children to participate in work inspired by the world that is created by the package so as not to emphasise unduly that it is intended for children who find reading difficult.



Selladore Tales graphics screen

At any level of competence, discussion will be required to solve the adventure (there is a crib in the teacher's book!) and a co-operative approach will be found the most productive despite the fact that the scenario is of modest length though challenging - appropriate to target users.

Alternatively, the program - with its ability to save and restart previous sessions - could form the basis for small group work on an ongoing basis, even as part of a specific reading programme for targeted children - so much a day, if that

Continued on page 55

Numerical Notations

by David Lowndes Williams

INTRODUCTION

I often use my Beeb to interpret data. When I do it's a bit disheartening when, after 10 minutes of number crunching, I get some answer like 1.5999999E-5. It doesn't look very good, and I have to think for a while to understand what it means. The program presented here contains a few procedures which will output numbers in a clearer way. It outputs numbers in "standard form" or "engineering form", using superscripted indices. You will probably find these facilities on your pocket calculator.

A NOTE ABOUT NOTATIONS

The number 16000 (sixteen thousand) may be represented, in standard form, as 1.6×10^4 , i.e. 1.6 times 10000 ("x" here represents multiplication). It may also be represented, in engineering form, as 16×10^3 . The important thing about standard form is the number on the left of the decimal point lies between 1 and 9. In engineering form the number on the left of the decimal point lies between 1 and 999, in such a way that the number to the right of the ten is a multiple of 3 (see program output). This is of use when choosing the type of units (e.g. m or km).

HOW TO USE THE PROGRAM

To output a number in standard form use PROCstandard(num,P%,D%,T%), and for output in engineering form use PROCengineering(num,P%,D%,T%), where num is the number that you want to print. If P% is set to TRUE then the output is to be sent to the printer. This is so that the computer can output the superscript printer codes when necessary. D% specifies the number of decimal places to be printed. T% tells the computer if it should print or suppress

trailing zeros. For example, if num=0.0012 and T%=TRUE then 1.200×10^{-3} would result. With T%=FALSE the number printed would be 1.2×10^{-3} . The best way to get familiar with the program and notations is to play about with it.

APPLICATIONS

PROCstandard and PROCengineering will be of use to output results from your own programs, and the ancillary procedures may be useful in other situations also. PROCsuper will be of use whenever superscript text is required. FNround is highly versatile; it is similar to Basic's INT() command, but rounds up instead of just truncating the remainder of a number (see also this month's Hints). PROCform will give the standard and engineering forms of numbers, outputting via s(0), s(1), e(0) and e(1).

Anybody who sets their BBC practical, numerical tasks, should find these procedures useful. They will improve the presentation of your data and will make numerical presentations look far more professional.

```
10 REM Program Numbers
20 REM Version B1.0
30 REM Author David Lowndes Williams
40 REM BEEBUG October 1991
50 REM Program subject to copyright
60 :
100 ON ERROR GOTO 230
110 MODE3:VDU19,0,4,0,0,0
120 DIMs(1):DIME(1)
130 :
140 PRINTTAB(2);"BASIC";TAB(20);"Stand
ard";TAB(35);"Engineering"
150 initial=7
160 FOR loop=-5 TO 10
```



```

170 number=initial*10^loop
180 PRINT TAB(2);number;TAB(20);:PROCS
tandard(number,FALSE,3,FALSE)
190 PRINT;TAB(35);:PROCEngineering(num
ber,FALSE,3,FALSE)
200 PRINT:NEXT
210 END
220 :
230 REPORT:PRINT" at line ";ERL:END
240 :
1000 DEF PROCform(number)
1010 LOCAL s%,l%,e%,mantissa
1020 IF number=0 s(0)=0:s(1)=0:e(0)=0:e
(1)=0:ENDPROC
1030 l%=LOG(ABS(number)):e%=INT(l%/3)*3
1040 mantissa=number*10^(-e%)
1050 IF ABS(mantissa)<1 e%=e%-3:mantiss
a=mantissa*1000
1060 s(0)=number*10^(-l%):s(1)=l%
1070 IF ABS(s(0))<1 s(1)=s(1)-SGN(s(0))
:s(0)=s(0)*(10^SGN(s(0)))
1080 e(0)=mantissa:e(1)=e%
1090 ENDPROC
1100 :
1110 DEF FNround(n)
1120 LOCAL n1,n2
1130 n1=INT(n):n2=n1+1
1140 IF ABS(n-n1)<ABS(n2-n) =n1 ELSE =n
2
1150 :
1160 DEF PROCsuper(text$,P%)
1170 LOCAL d%,X%,Y%,z%
1180 IF P%=TRUE VDU2,1,27,1,83,1,0:PROC
printer(text$):VDU1,27,1,84,3
1190 d%=&70
1200 FOR z%=1 TO LENText$
1210 ?d%=ASC(MID$(text$,z%,1))
1220 X%=d%:Y%=d% DIV 256:A%=&A:CALL&FFF
1
1230 VDU23,224,?(d%+1),?(d%+3),?(d%+4),
?(d%+5),?(d%+7),0,0,0
1240 PRINT;CHR$224;:NEXT
1250 IF P%=TRUE VDU2
1260 ENDPROC
1270 :
1280 DEF PROCprinter(text$)
1290 LOCAL d%,z%
1300 FOR z%=1 TO LENText$
1310 d%=ASC(MID$(text$,z%,1))

```

```

1320 VDU1,d%:NEXT z%
1330 ENDPROC
1340 :
1350 DEF FNstring(n,D%,T%)
1360 LOCAL t$,j%,h,i,d
1370 n=FNround(n*10^D%)*10^-D%
1380 t$="":IF n<0 t$=t$+"-":n=-n
1390 FOR j%=5 TO -D% STEP-1
1400 IF j%=-1:IF RIGHT$(t$,1)="" OR RIG
HT$(t$,1)="-" t$=t$+"0":GOTO1420
1410 IF j%=-1 t$=t$+"."
1420 h=n*10^-j%:i=INT(h):d=h-i
1430 IF d>0.99999 i=i+1
1440 h=i MOD 10
1450 IF h=0:IF RIGHT$(t$,1)="" OR RIGHT
$(t$,1)="-" GOTO1470
1460 t$=t$+CHR$(48+h)
1470 NEXT
1480 IF T%=FALSE t$=FNsurpress(t$)
1490 =t$
1500 :
1510 DEF FNsurpress(text$)
1520 LOCAL l%
1530 REPEAT:l%=LEN(text$)
1540 IF RIGHT$(text$,1)="0" text$=LEFT$(
text$,l%-1)
1550 UNTIL l%=0 OR RIGHT$(text$,1)<>"0"
1560 l%=LEN(text$)
1570 IF RIGHT$(text$,1)=". " text$=LEFT$(
text$,l%-1)
1580 =text$
1590 :
1600 DEF PROCstandard(num,P%,D%,T%)
1610 PROCform(num)
1620 man$=FNstring(s(0),D%,T%):PRINT;ma
n$;
1630 IF s(1)=0 ENDPROC
1640 PRINT"x10^";
1650 exp$=STR$(s(1)):PROCsuper(exp$,P%)
1660 ENDPROC
1670 :
1680 DEF PROCEngineering(num,P%,D%,T%)
1690 PROCform(num)
1700 man$=FNstring(e(0),D%,T%):PRINT;ma
n$;
1710 IF e(1)=0 ENDPROC
1720 PRINT"x10^";
1730 exp$=STR$(e(1)):PROCsuper(exp$,P%)
1740 ENDPROC

```





BEEBUG Function/Procedure Library (6)

by Andrew Rowland and David Stevens

This month's additions to the library concentrate on a group of sounds and associated envelopes provided and described by Andrew Rowland. Because of their nature they are documented differently from the style used previously. We have also included this time a procedure from David Stevens for plotting dotted lines at any angle on the screen.

Further contributions to the library are always welcome.

SOUND LIBRARY

Probably the least understood area of the BBC micro is sound. Nothing can replace the experience gained spending hours tinkering with the SOUND and ENVELOPE commands. Often, however, you only need a simple effect to accompany a game, and someone else may know just what you need. So if you have a useful sound effect or a good imitation of a musical instrument, why not contribute it to the library?

Here are a few to get the ball rolling. As the procedures are mostly self explanatory, I will give only the briefest of notes.

PROCenvelopes

It is good practice to define envelopes at the beginning of a program. Note that this procedure uses part of page &900 to store envelopes 5-8.

PROCTing(pitch)

Bell sound - try *pitch* 150-250

PROCclink(pitch)

Metallic sound - try *pitch* 180-220

PROCExplore

PROCcrash

PROCTrimphone

Rings until a key is pressed. The key may be detected in the usual way with GET etc.

PROCCroak

PROCFrog

PROCbounce

Sound of hard object falling on wooden floor.

PROCTwang(pitch)

Rubber band - try *pitch* 100-150

PROCGuitar(pitch)

PROCwait(t%)

Delay in 1/20s (the same period used for the duration in SOUND).

Routine:	Dotlin
Type:	PROCEDURE
Syntax:	PROCdotlin(x%,y%,x1%,y1%,d%)
Purpose:	Plots even dotted line at any angle
Parameters:	x%,y% Start co-ordinates x1%,y1% End co-ordinates d% Dot spacing
Notes:	Try values of d% between 20 and 48 (any graphics mode)
Related:	None

```

10 REPEAT
20 MODE0
30 INPUTTAB(1,5)"Dot spacing: "d%
40 X%=200:Y%=100:MX%=1000:MY%=800
50 PROCdotlin(X%,Y%,X%,MY%,d%)
60 PROCdotlin(X%,Y%,MX%/2,MY%,d%)
70 PROCdotlin(X%,Y%,MX%,MY%,d%)
80 PROCdotlin(X%,Y%,MX%,MY%/2,d%)
90 PROCdotlin(X%,Y%,MX%,Y%,d%)
100 P.TAB(5,30)"Any key for another go"
110 IF GET:UNTIL FALSE

```


BEEBUG Function/Procedure Library

```
26500 REM Sound Library
26510 :
26520 DEF PROCenvelopes
26530 bell=1:bomb=2:crash=3:phone=4:clink=5
26540 rubber_band=6:guitar=7:guitar2=8
26550 ENVELOPE bell,1,0,0,0,0,0,126,-1,0,-1,126,0
26560 ENVELOPE clink,1,0,0,0,0,0,126,-10,0,-20,126,0
26570 ENVELOPE bomb,1,0,0,0,0,0,126,-1,0,-1,126,126
26580 ENVELOPE crash,1,0,0,0,0,0,127,-1,-1,-1,126,0
26590 ENVELOPE phone,2,33,-33,33,2,2,2,127,0,0,-20,126,0
26600 ENVELOPE rubber_band,1,0,0,0,0,0,126,-6,-10,-5,126,0
26610 ENVELOPE guitar,1,0,0,0,0,0,126,-1,0,-1,126,20
26620 ENVELOPE guitar2,1,0,0,0,0,0,50,-1,-1,-1,50,10
26630 ENDPROC
26640 :
26650 DEF PROCting(pitch):REM 150-200
26660 SOUND 1,bell,pitch,5
26670 ENDPROC
26680 :
26690 DEF PROCclink(pitch):REM 180-220
26700 SOUND 1,clink,pitch,1
26710 ENDPROC
26720 :
26730 DEF PROCexplode
26740 SOUND 0,bomb,6,9
26750 ENDPROC
26760 :
26770 DEF PROCcrash
26780 SOUND 0,crash,20,20
26790 ENDPROC
26800 :
26810 DEF PROCtriphone
26820 REPEAT
26830 SOUND 1,phone,76,6:PROCwait(12):SOUND 1,phone,76,8
26840 PROCwait(40):UNTIL NOT INKEY-129
26850 ENDPROC
```

```
26860 :
26870 DEF PROCcroak
26880 PROCfrog:PROCwait(10):PROCfrog
26890 ENDPROC
26900 :
26910 DEF PROCfrog
26920 SOUND 1,0,0,0
26930 SOUND 0,-15,3,5
26940 FOR P=0 TO 150 STEP 30
26950 SOUND 1,0,P,1
26960 NEXT
26970 ENDPROC
26980 :
26990 DEF PROCbounce
27000 SOUND 0,bell,3,1:SOUND 1,0,1,1
27010 ENDPROC
27020 :
27030 DEF PROCtwang(pitch):REM 100-150
27040 SOUND 0,rubber_band,3,10:SOUND 1,0,pitch,10
27050 ENDPROC
27060 :
27070 DEF PROCguitar(pitch)
27080 SOUND &211,guitar,pitch,20:SOUND &212,guitar2,pitch+48,20
27090 SOUND &213,guitar2,pitch-48,20
27100 ENDPROC
27110 :
27120 DEF PROCwait(t%):REM 1/20s
27130 LOCAL T%:T%=TIME:REPEAT UNTIL INKEY-129
27140 REPEAT UNTIL TIME>T%+t%*5 OR NOT INKEY-129
27150 ENDPROC
27160 :
27170 REM Dotted Line
27180 :
27190 DEF PROCdotlin(x%,y%,x1%,y1%,d%)
27200 LOCAL b,p,h,l,dx,dy
27210 b=x1%-x%:p=y1%-y%:h=SQR(b*b+p*p)
27220 l=h/d%:dx=0:dy=0
27230 IF b<>0 dx=b/l
27240 IF p<>0 dy=p/l
27250 FOR j%=1 TO l
27260 PLOT69,x%+(j%-1)*dx,y%+(j%-1)*dy
27270 NEXT:ENDPROC
```

Hexagons

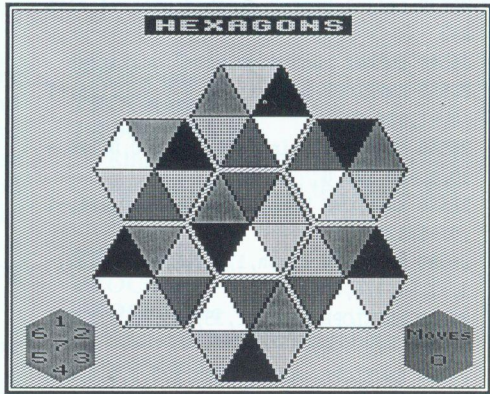
A game by Anthony Birch

Like many games published in BEEBUG, Hexagons is designed more to tax the brain than the manual dexterity of your fingers. The screen display shows a total of seven regular hexagons arranged symmetrically on the screen. The six segments of each hexagon are shaded in six different colours.

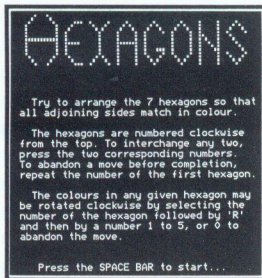
The object of the game is quite simple - to rearrange the hexagons so that the colours of segments with a common boundary are of the same colour. This can be achieved by either swapping the positions of any two hexagons, or by rotating a specified hexagon through one or more segments.

The program is in two parts, with the first part detailing the sound and character definitions, and part two playing the game itself. Type in the program in listing 1 and save this as *Hexagon*; then type in listing 2 and save this as *Hex2*.

Hexagon is the program which should be run first, and this automatically chains Hex2. The program starts by displaying instructions for the game, before the hexagons themselves appear. These are numbered from 1 to 7 (as shown on screen). Any two hexagons may be exchanged by typing in their numbers. A hexagon can be rotated (always clockwise), by entering its number, followed by 'R', and then the number of



Hexagons game display



Game instructions

segments by which you want to rotate it. You can stop playing at any time by pressing Escape, which returns you to the instruction screen; pressing Escape there terminates the game.

All that remains is to work out the moves needed to achieve the game's aims. And on the way, the computer will keep a count (on screen) of the number of moves which you take. Have fun!

Listing 1

```
10 REM Program Hexagon
20 REM Version B1.1
30 REM Author Anthony Birch
40 REM BEEBUG October 1991
50 REM Program subject to copyright
60 :
100 @%=0:*FX4,1
110 *FX11,0
120 ENVELOPE1,1,0,0,0,1,1,1,80,255,255
,253,100,10
130 ENVELOPE2,1,0,0,0,2,2,2,30,0,0,255
,80,1
140 VDU23,230,0,14,17,17,17,17,14
```



```

150 VDU23,231,0,4,12,4,4,4,4,14
160 VDU23,232,0,14,17,1,2,12,16,31
170 VDU23,233,0,14,17,1,6,1,17,14
180 VDU23,234,0,2,6,10,18,31,2,2
190 VDU23,235,0,31,16,16,30,1,17,14
200 VDU23,236,0,6,8,16,30,17,17,14
210 VDU23,237,0,31,1,2,4,8,8,8
220 VDU23,238,0,14,17,17,14,17,17,14
230 VDU23,239,0,14,17,17,15,1,2,12
240 VDU23,240,0,68,108,84,85,69,69,68
250 VDU23,241,0,0,0,149,85,85,73,137
260 VDU23,242,0,0,0,204,16,156,4,216
270 :
280 J=&A80:FOR I=0 TO 31:READJ?I:NEXT
290 FOR I=0 TO 2 STEP 2:P%=&900
300 [OPTI
310 .fillinit
320 LDA#0:STA&73:TAY
330 ASL&75:ASL&75:DEC&75:DEC&76
340 ASL&76:ASL&76:LDA#15:STA&74
350 .main
360 JSRright:JSRleft
370 DEC&74:BNEmain:JSRright
380 RTS
390 .right:JSRfirstright
400 .r1
410 LDA(&70),Y:LDX&75:CMP&A80,X:BEQr3
420 DEX:CMP&A80,X:BEQr4
430 JSRline:LDA&72:BEQr2:JSRsub
440 .r2:RTS
450 .r3
460 LDA#3:JSRprint:LDX#8:JSRadd
470 JMPrl
480 .r4
490 LDA#2:JSRprint:JSRline:RTS
500 .firstright
510 LDA(&70),Y:LDX&75
520 DEX:DEX:CMP&A80,X:BNEfr1
530 LDA#1:JSRprint
540 .fr1
550 INX:INX:CMP&A80,X:BNEfr2
560 LDA#3:JSRprint
570 .fr2:LDX#8:JSRadd:RTS
580 .left:JSRfirstleft
590 .l1
600 LDA(&70),Y:LDX&75:CMP&A80,X:BEQl3
610 DEX:DEX:CMP&A80,X:BEQl4
620 JSRline:LDA&72:BEQl2

```

```

630 LDX#8:JSRadd
640 .l2:RTS
650 .l3:LDA#3:JSRprint:JSRsub:JMPl1
660 .l4:LDA#1:JSRprint:JSRline:RTS
670 .firstleft
680 LDA(&70),Y:LDX&75
690 DEX:CMP&A80,X:BNEfl1
700 LDA#2:JSRprint
710 .fl1
720 INX:CMP&A80,X:BNEfl2
730 LDA#3:JSRprint
740 .fl2:JSRsub:RTS
750 .line
760 LDA&70:AND#7:CMP#7:BEQln
770 LDX#1:JSRadder:RTS
780 .ln
790 LDX#&D3:JSRadder
800 JSRadder:JSRadder:RTS
810 .add
820 LDA&73:CMP#1:BEQa1
830 JSRadder:LDA#1:STA&73
840 .a1:RTS
850 .adder
860 TXA:CLC:ADC&70:STA&70
870 LDA#0:ADC&71:STA&71
880 RTS
890 .sub
900 LDA&73:CMP#2:BEQs1
910 LDA&70:SEC:SEC#8:STA&70
920 LDA&71:SBC#0:STA&71
930 LDA#2:STA&73
940 .s1:RTS
950 .print
960 CLC:ADC&76:TAX
970 LDA&A80,X:STA(&70),Y
980 LDA#0:STA&73
990 RTS
1000 .back
1010 LDA#&30:STA&71
1020 LDA#0:STA&70:TAY
1030 .b1
1040 TYA:AND#1:BEQb2
1050 LDA#7:JMPb3
1060 .b2:LDA#&B
1070 .b3
1080 STA(&70),Y:INY:BNEb1
1090 INC&71:LDA&71:CMP#&80:BNEb1
1100 RTS

```

Hexagons

```
1110 ]
1120 NEXT
1130 :
1140 B%=back:F%=fillinit
1150 CHAIN"Hex2"
1160 :
1170 DATA 0,1,2,3,0,4,8,12,0,5,10,15,0,
16,32,48,0,17,34,51,0,20,40,60,0,21,42,6
3,70,73,78,73
```

Listing 2

```
10 REM Program Hex2
20 REM Author Anthony Birch
30 REM Program subject to copyright
40 :
100 DIM M 84,A(7),H(24),N(14),O(6)
110 REPEAT
120 MODE7:VDU23,1,0;0;0;0;
130 ON ERROR GOTO 2850
140 PROCinstruct:PROCspace
150 MODE2:VDU23,1,0;0;0;0;
160 ON ERROR GOTO 2840
170 RESTORE2680
180 CALL B%
190 PROCscr:*FX15,0
200 REPEAT
210 HA=GET-48
220 IF HA<1 OR HA>7 SOUND4,1,50,4:GOTO
210
230 SOUND1,-12,150,2
240 GCOL0,8:PROCnum(HA)
250 HB=GET-48
260 IF HB=34 OR HB=66 THEN 380
270 IF HB<1 OR HB>7 SOUND4,1,50,4:GOTO
250
280 SOUND1,-12,200,2
290 IF HB=34 THEN 390
300 GCOL0,0:PROCnum(HA)
310 GCOL0,8:PROCnum(HB)
320 PROCswap
330 H=HA:PROCfill:H=HB:PROCfill
340 IF HA<HB PROCcount(1)
350 IF N=12 THEN 510
360 GCOL0,0:PROCnum(HB)
370 UNTIL FALSE
380 SOUND1,-12,200,2
390 GCOL0,6:PROCnum(HA)
400 HR=GET-48
```

```
410 IF HR<0 OR HR>5 SOUND4,1,50,4:GOTO
400
420 PROCrotate
430 FOR I=1 TO 40
440 SOUND1,-12,100+I,0
450 NEXT
460 H=HA:PROCfill
470 IF HR>0 PROCcount(1)
480 IF N=12 THEN 510
490 GCOL0,0:PROCnum(HA)
500 UNTIL FALSE
510 MODE7:VDU23;11,0;0;0;0;
520 PRINT:PROctitle
530 IF moves>12m$="WELL DONE" ELSE m$=
"EXCELLENT!"
540 FOR I=11 TO 12
550 PRINTTAB(12,I)CHR$130;CHR$141;m$
560 NEXT
570 PRINT"CHR$131"You've completed Hex
agons in ";moves;" moves"
580 PROCspace
590 UNTIL FALSE
600 :
1000 DEF PROCscr
1010 VDU24,364;952;904;1000;
1020 GCOL0,130:CLG:VDU26
1030 COLOUR130:COLOUR4
1040 PRINTTAB(6,1)"HEXAGONS"
1050 FOR I=1 TO 7:READX,Y
1060 VDU29,X;Y;
1070 GCOL0,7:P=85:PROChex
1080 GCOL0,0:P=5:PROChex
1090 NEXT:PROCinit
1100 FOR H=1 TO 7:PROCfill:NEXT
1110 GCOL0,1:VDU29,142;142;:PROCexr
1120 VDU29,1148;142;:PROCexr
1130 VDU29,0;0;5:GCOL0,0
1140 FOR I=1 TO 7:PROCnum(I):NEXT
1150 MOVE1060,188
1160 PRINTCHR$240;CHR$241;CHR$242
1170 MOVE1108,120:PRINTCHR$230
1180 ENDPROC
1190 :
1200 DEF PROctitle
1210 RESTORE
1220 FOR C=1 TO 5
1230 PRINTCHR$150;CHR$154;
1240 FOR J=1 TO 37:READ K
```



```

1250 PRINTCHR$(K+160);
1260 NEXT:PRINT:NEXT:PRINT''
1270 ENDPROC
1280 :
1290 DEF PROCchex
1300 MOVE-80,132
1310 DRAW-160,0:PLOTP,-80,-132
1320 DRAW80,-132:PLOTP,-80,132
1330 MOVE80,132:DRAW160,0
1340 PLOTP,80,-132:DRAW-80,132
1350 PLOTP,80,132:DRAW-80,-132
1360 MOVE-160,0:DRAW160,0
1370 ENDPROC
1380 :
1390 DEF PROCchexr
1400 GCOL0,1:MOVE-88,48
1410 MOVE0,96:PLOT85,84,48
1420 MOVE-88,48:PLOT85,-88,-48
1430 MOVE84,48:PLOT85,84,-48
1440 MOVE0,-96:PLOT85,-88,-48
1450 ENDPROC
1460 :
1470 DEF PROCmix
1480 FOR S=0 TO 6:R=RND(7)-1
1490 FOR I=1 TO 6:T=M?(S*6+I):M?(S*6+I)
=M?(R*6+I):M?(R*6+I)=T
1500 NEXT:NEXT
1510 FOR HA=1 TO 7:HR=RND(5):PROCrotate
1520 NEXT
1530 ENDPROC
1540 :
1550 DEF PROCfill
1560 A=A(H)
1570 FOR J=1 TO 6
1580 A=A+O(J)
1590 IF A=&4540 A=&47B8
1600 IF A=&476F A=&44F7
1610 IF A=&5C68 A=&5EE0
1620 IF A=&5B68 A=&5DE0
1630 ?&70=A MOD256:??&71=A DIV256
1640 ?&72=J MOD2
1650 ?&75=M?((H+6)*6+J)
1660 ?&76=M?((H-1)*6+J)
1670 CALL F%
1680 NEXT
1690 ENDPROC
1700 :
1710 DEF PROCnum(H)

1720 IF H>7 ENDPROC
1730 MOVE N(2*H-1),N(2*H)
1740 PRINTCHR$(H+230)
1750 ENDPROC
1760 :
1770 DEF PROCswap
1780 FOR I=1 TO 6
1790 M?((HA+6)*6+I)=M?((HA-1)*6+I)
1800 M?((HB+6)*6+I)=M?((HB-1)*6+I)
1810 M?((HA-1)*6+I)=M?((HB+6)*6+I)
1820 M?((HB-1)*6+I)=M?((HA+6)*6+I)
1830 NEXT
1840 ENDPROC
1850 :
1860 DEF PROCrotate
1870 FOR I=1 TO 6
1880 M?((HA+6)*6+I)=M?((HA-1)*6+I)
1890 NEXT
1900 FOR I=1 TO 6
1910 J=I+HR
1920 IF J>6 J=J-6
1930 M?((HA-1)*6+J)=M?((HA+6)*6+I)
1940 NEXT
1950 ENDPROC
1960 :
1970 DEF PROCcount(CL)
1980 GCOL0,CL:PROCmoves
1990 IF N=12 ENDPROC
2000 moves=moves+1:GCOL0,0
2010 PROCmoves:PROCcheck
2020 ENDPROC
2030 :
2040 DEF PROCmoves
2050 IF moves<10 MOVE1108,120:PRINTCHR$(
moves+230)
2060 IF moves>9 AND moves<100 MOVE1080,
120:PRINTCHR$((moves DIV10)+230):MOVE112
8,120:PRINTCHR$((moves MOD10)+230)
2070 IF moves>99 MOVE1060,120:PRINTCHR$(
(moves DIV100)+230):MOVE1108,120:PRINTC
HR$(((moves DIV10)MOD10)+230):MOVE1156,1
20:PRINTCHR$((moves MOD10)+230)
2080 ENDPROC
2090 :
2100 DEF PROCcheck
2110 N=0
2120 FOR I=1 TO 12
2130 IF M?H(I)=M?H(I+12)N=N+1

```

Hexagons

```
2140 NEXT
2150 IF N=12 PROCsuccess
2160 ENDPROC
2170 :
2180 DEF PROCsuccess
2190 FOR I=0TO200STEP6
2200 FOR J=1TO100:NEXT
2210 SOUND1,2,I,1
2220 SOUND2,2,I+32,1
2230 SOUND3,2,I+48,1
2240 NEXT
2250 GCOL0,0:PROCnum(HA):PROCnum(HB)
2260 PROCcount(8):I=INKEY(500)
2270 ENDPROC
2280 :
2290 DEF PROCinit
2300 N=0:moves=0
2310 FOR I=1 TO 42:READ M?I:NEXT
2320 FOR I=1 TO 24:READH(I):NEXT
2330 PROCmix
2340 FOR I=43 TO 84:M?I=7:NEXT
2350 FOR H=1 TO 7:READA$
2360 A(H)=EVAL("&" + A$):NEXT
2370 FOR I=1 TO 6:READO(I):NEXT
2380 FOR I=1 TO 14:READN(I):NEXT
2390 ENDPROC
2400 :
2410 DEF PROCinstruct
2420 PROCtitle
2430 PRINTCHR$130;" Try to arrange the
7 hexagons so that";
2440 PRINTCHR$130;"all adjoining sides
match in colour."
2450 PRINT'CHR$131;" The hexagons are
numbered clockwise"
2460 PRINTCHR$131;"from the top. To int
erchange any two,"
2470 PRINTCHR$131;"press the two corres
ponding numbers."
2480 PRINTCHR$131;"To abandon a move be
fore completion,"
2490 PRINTCHR$131;"repeat the number of
the first hexagon.";
2500 PRINT'CHR$130;" The colours in an
y given hexagon may"
2510 PRINTCHR$130;"be rotated clockwise
by selecting the"
2520 PRINTCHR$130;"number of the hexago
n followed by 'R'"
```

```
2530 PRINTCHR$130;"and then by a number
1 to 5, or 0 to"
2540 PRINTCHR$130;"abandon the move."
2550 ENDPROC
2560 :
2570 DEF PROCspace
2580 PRINT' 'CHR$132;" Press the SPACE
BAR to start...";
2590 *FX15,0
2600 REPEAT UNTIL GET=32
2610 ENDPROC
2620 :
2630 DATA64,6,0,2,68,0,0,0,80,16,64,0,0
,64,0,0,64,0,0,0,16,0,0,0,64,0,0,64,0
,0,64,0,0,64,0,0
2640 DATA21,0,0,0,21,64,6,0,0,0,9,24,
1,0,24,1,9,16,64,6,2,68,0,24,1,9,16,74,0
,0,74,0,24,1,9,16
2650 DATA29,12,12,12,12,21,74,80,80,0,0
,0,21,0,0,85,80,80,21,74,0,0,0,21,0,0,
21,74,9,16,74,0,2,68,0,0
2660 DATA21,0,0,0,0,21,74,0,0,0,0,21,
0,0,21,0,0,21,74,0,8,76,0,21,0,0,21,74,0
,2,78,0,16,0,9,16
2670 DATA2,68,0,64,6,0,0,9,80,16,64,6,2
,68,0,21,0,0,21,0,9,24,1,0,2,68,6,0,74,0
,0,74,0,2,68,6,0
2680 DATA642,740,898,600,898,320,642
2690 DATA180,386,320,386,600,642,460
2700 DATA3,5,2,1,4,6,3,5,4,1,6,2
2710 DATA1,5,4,6,3,2,4,3,2,1,6,5
2720 DATA6,3,5,4,1,2,2,4,5,6,3,1
2730 DATA1,6,2,4,3,5
2740 DATA4,11,18,19,26,33
2750 DATA3,10,17,24,25,32
2760 DATA37,38,39,40,41,42
2770 DATA12,13,20,27,34,5
2780 DATA3B1E,4819,5C1F
2790 DATA681A,5B1F,4719,519C
2800 DATA0,73,2528,-7,-73,-2528
2810 DATA100,220,156,188,156,124
2820 DATA100,92,40,124,40,188,100,156
2830 :
2840 IF ERR=17 THEN GOTO 110
2850 MODE7:IF ERR<>17 REPORT:PRINT" at
";ERL
2860 *FX4,0
2870 *FX12,0
2880 END
```

B

is your preferred style. It is certainly flexible enough.

The teacher's book explains how the text has been subjected to two respected reading level tests. Acknowledging that this is a controversial area, Sherston assess the material to be appropriate to pupils whose reading age is between 7 and 9 years. This should be taken as a rough guide and factors such as motivation and pupils' involvement in the subject matter will have a bearing on its success or failure too.

The activity sheets each cover a specific part of the story and are of four types: cloze procedure, word searches (left to right horizontally and diagonally downwards and vertically downwards only), crosswords and sequencing. For the lazy or busy teacher the solutions to most of these activities, too, are provided in the teacher's manual.

All in all this is a good selection of techniques and they could provide in themselves more activity - the

compilation by the pupils of word searches for their friends to solve, for example.

There are sheets to stimulate free writing, and several useful and feasible suggestions for use of the text files - loading them into a word processor and searching for the actual characters' names to replace them with those of your own pupils, for example. All of these are sound and educationally valid. There is a short list of children's books likely to prove relevant and interesting in supplementing Selladore Tales.

CONCLUSIONS

Sherston has established a name for reliable, well researched and well written software for schools. Selladore Tales hardly breaks new ground. What it does do is address a specific area of school experience, language development, in a sensible, down-to-earth and accessible way. It has been done well, attractively presented and thoroughly researched. It is also sensibly priced. Recommended. **B**

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

JUMBO CROSSWORDS (Vol.9 No.7)

A number of readers have asked if this program could be modified to allow the use of Spellmaster for locating and checking suitable words using Spellmaster's *CROSSWORD command. Richard Jones, who wrote the original program, has provided the necessary changes/additions.

Change the following lines as shown:

```
220 PRINT TAB(0,0)"Use: Cursor Keys,  
Space or Delete. N to Number, D to Dump,  
Q to Quit, S to Spell";
```

```
320UNTIL INKEY(-51) OR INKEY(-17) OR  
INKEY(-86) OR INKEY(-82)
```

```
330 IF INKEY(-17) M$="Quit" ELSE IF  
INKEY(-51) M$="Dump to printer" ELSE IF  
INKEY(-86) M$="Number squares" ELSE IF  
INKEY(-82) M$="Spellmaster"
```

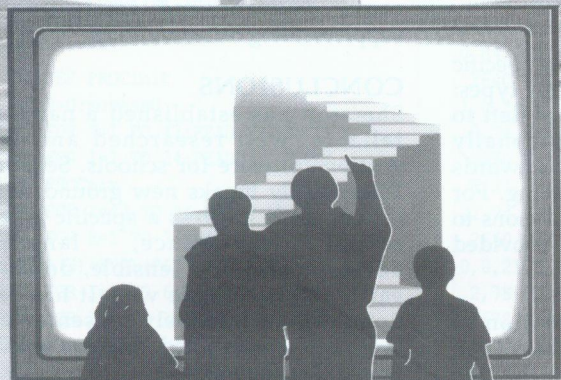
```
420 IF C$="D" PROCdump ELSE IF C$="N"  
PROCNnumber ELSE IF C$="S" PROCspell
```

and add the following lines:

```
2590:  
2600 DEF PROCspell  
2610 VDU28,64,31,79,6,14,12  
2620 PRINT"Search: ":*FX21  
2630 INPUT A$:IF A$="" GOTO 2670  
2640 ON ERROR PRINT"None found":  
PROCcontinue:GOTO150  
2650 PRINT"Press Shift for more"  
2660 OSCLI("Crossword"+A$)  
2670 PROCcontinue  
2680:  
2690 DEF PROCcontinue  
2700 PRINT"Press a key to continue"  
2710 IF GET  
2720 VDU12,15,26  
2730 ENDPROC
```


THE MICRO USER

Volume 9 Number 7 September 1991 £1.50



**THE
MICRO
USER**

**YOUR £400
WORTH OF
MONEY-OFF
VOUCHERS**

SAVE A BOMB!
With £400 worth
of money-off
vouchers

For ALL users of the BBC Micro, Master, Archimedes series, A3000, Electron

EUROPRESS
PUBLICATIONS

COMPUTE!

8 Extra
pages
for all 8-bit
beginners

PLUS

8 Extra
pages
for new
Arc owners

PLUS

FREEBIES
**1,500 FREE
DISCS OFFER**

PLUS

Every one of your
regular monthly
features in this
bigger, brighter

**MICRO
USER**



**OCTOBER ISSUE ON SALE
12th SEPTEMBER**

HINTS HINTS HINTS HINTS HINTS

and tips and tips and tips and tips and tips

This month we have a varied collection of hints showing that there are still new things to learn, and useful reminders of perhaps forgotten techniques. Further hints and tips for this page are always welcome.

USING FUNCTION KEY 10

Roger Smith

The reference in *Passing Information via Function Keys* in BEEBUG Vol.10 No.3 to the fact that the Break key (function key 10) cannot be used in this way is not entirely correct. The Break key does not produce a character code at all, but is connected directly to the 6502 CPU's reset input. It is the operating system, as part of the initialisation sequence after a Break, which inserts the programmed contents of function key 10 into the input buffer (using, in effect *FX202 (&CA), not *FX138 (&8A) as might be expected - in the default state characters &C0- &CF activate the function keys in the same way as characters &80-&8F).

All this means is that using *FX138 is a perfectly safe way of passing information via the mythical function key 10, and is thus no different from using the other function keys in this respect.

FLOATING POINT ROUNDING

Ben Avison

In Basic programming it is not uncommon to need to find the 'closest' integer to a floating point value. The function INT() is a common method for doing this, and returns the next lowest integer (it rounds down). Simply assigning the floating point value to an integer variable will give the next lowest integer for positive numbers, or the next highest integer for negative numbers (it rounds to zero). There is also a third, undocumented form of rounding:

`-INT(-value)`

will round up (those are minus signs).

Some examples of the results of using these three methods are shown in table 1 (where val is a real variable).

val	N% = INT(val)	N% = val	N% = -INT(-val)
+2.0	+2	+2	+2
+1.9	+1	+1	+2
+1.1	+1	+1	+2
0.0	0	0	0
-1.1	-2	-1	-1
-1.9	-2	-1	-1
-2.0	-2	-2	-2

Table 1. Results of rounding

FORMATTING HELP IN EDIT

Andrew Rowland

Master users are familiar with the extensive on-screen help which Edit provides, but one area always had me reaching for the manual - the formatting commands which turn Edit into a powerful word processor. But help is at hand: press f8 (Print text) and then H (for Help), and a full list of the formatting commands is displayed, with their syntax and default values where appropriate.

When printing from Edit programs which contain assembler, type the following at the beginning of the file:

`.cc@`

It will stop labels being possibly mistaken for a formatting command.

FUNCTION KEYS IN VIEW

David Holton

When in command mode, the function keys in View normally have no role to play. However, you can program them to do all sorts of useful things for you, without affecting their functions in edit mode. I use my !BOOT file to set them up to change drives, save, *CAT, and control my printer buffer. Other handy functions are *ACCESS, *.* and *COPY.

B

RISC USER

The Archimedes Magazine & Support Group

Now in its fourth 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 magazine or extend their 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:

MEGABYTES FOR ALL

An overview of hard drive options, including the latest SCSI and IDE devices.

ACORN'S MULTI-TASKING PC EMULATOR

A review of this powerful multi-tasking PC emulator.

PROCEDURE LIBRARY MANAGER

An extremely useful application, which provides a library to store and manage frequently used procedures and functions.

UNIX FOR THE ARC

It is now possible to upgrade 400/1 and 500 series Archimedes to UNIX. An article about installing and running UNIX on the Arc.

DRAWFONT

A powerful application which provides the means to manipulate outline fonts in Draw.

PICTURE KIT

A look into this exciting new development for the Archimedes, which allows you to take photographs with Canon's Ion camera (which stores the image on disc), view them instantly on your computer screen and print them in colour or black and white.

DESKTOP FOLIO

A review of this colourful and sophisticated package, which combines a wordprocessor, basic DTP and interactive publishing in an easy-to-use system, primarily designed for the education market.

FORM DESIGNER

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

ARCHIMEDES CLIP ART

A look into some recent collections of clip art.

USING ANSI C

A series of articles on programming Desktop applications in C.

WP/DTP

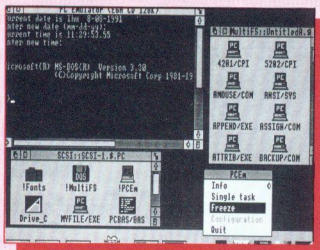
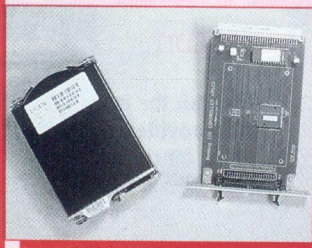
A regular column on using different DTP and WP packages. The latest offer is a simple text printer utility.

ARCADE

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

INTO THE ARC

A regular series for beginners.



SUBSCRIPTION DETAILS

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

Destination	Additional Cost
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



'EPSON COMPATIBILITY'

I was delighted to read your editorial on 'Epson compatibility' in Vol.10 No.4. In recent years I have been sadly disillusioned by the random use of the phrase 'Epson compatible'. The phrase should be 'Epson FX80 compatible' as I am sure you would know if you had tried to run some of the BEEBUG 'Epson compatible' programs on an MX80. My problem is that I have not been able to get a clear idea of the fundamental differences which make the MX a non-runner.

How about an article detailing the differences and possible ways round these to help those of us with different printers?

Peter Toft

First of all, it is worth noting that the phrase 'Epson compatible' is one in general use, no doubt coined a good few years ago by other manufacturers seeking to get a foothold in the then Epson dominated market. Our use of the phrase, when it has occurred, simply follows general usage.

However, there are different degrees of compatibility even between the various Epson models, and we are currently pursuing the suggestion of an article covering this topic.

The concept of printer drivers, used by View on the BBC micro and by the Archimedes range, overcomes this problem. The program 'talks' to a standard printer driver, which in turn is tailored to the user's printer.

MORE ECONOMICAL MOVEMENT

The Vol.10 No.4 issue of BEEBUG featured a program for drawing a picture using a long list of DRAW and MOVE statements. A neater alternative is to store the co-ordinates in DATA statements, as the following procedure for drawing a triangle shows:

```
100 MODE1
110 PROCgraphics
```

```
120 END
130 :
140 DEF PROCgraphics
150 REPEAT
160 READ functions$,x,y
170 IF function$="M" THEN MOVE x,y
180 IF function$="D" THEN DRAW x,y
190 UNTIL function$="E"
200 ENDPROC
210 :
220 DATA M,100,100,D,1180,100,D,640,92
4,D,100,100,E,0,0
```

Letters 'M' and 'D' denote MOVE and DRAW respectively, with the two co-ordinates following in each case. An 'E' at the end of the data (plus two dummy zero values for completeness) terminates the program.

Rowland Fraser

It is always worth looking for patterns in programs to avoid unnecessary repetition, and this is a useful example of the economy which can result.

NO GAMES PLEASE

"BEEBUG is the only magazine on the market which is not turned over to games but stays with sensible computing. It is the only magazine worth buying."

Tom Turner

"Put me firmly on the side of the 'No Games' lobby."

Peter Baldwin

These are just two of the comments received in response to the letter on magazine games content in Vol.10 No.4. I am sure that most readers, in all honesty, have enjoyed playing the occasional computer game, and games will continue to appear in BEEBUG from time to time. But rest assured that there is no danger of BEEBUG becoming less serious and turning into a games magazine - unless that really is what the majority of readers want.

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.

Spellcheck III ROM, Dictionary disc, instructions for M128 £12 (including c.o.d. & postage). Tel. (0673) 60892.

WANTED: BBC text adventure on disc or cassette. **FOR SALE:** All other types of BBC games and some utilities on discs and cassettes, (a number are Master or Electron compatible), all are originals and in good condition. Also can anyone provide me with any information on clubs for BBC text adventure please. Tel. (0438) 361099.

First Word Plus (release II) unopened £35, Genesis unused £20, Interdicator II as new £20, The Wimp Game as new £9. Tel. 081-841 1463 eves/wk/ends.

ADT ROM Interword ROM £20, BEEBUG Master ROM £12, Elite disc £8, Impact, Ninja, Maniac's Diary, Calligraphy, Zoom, Knitware Designer and Modem Master all £3 each. Tel. (0475) 38502.

M128 plus discs, manual and printer lead £250 or offers, also Overview complete with all documentation, offers. Tel. (0274) 878881.

BBC B 1770/8271 DFS fitted with 256k RAM board and wordwise plus £200, twin Teac 800k 40/80 switchable drives with PSU £120, Star Gemini 10x printer £80, Music 5000 £40, Acorn TText adaptor £35, Prism 2000 modem £15, loads of discs, may consider splitting or make me an offer for the lot. Tel. 091-262 7764.

512 board fitted in Watford co-pro adaptor together with GEM mouse and software, hardly used £180 o.n.o. also Problem Solver v1.2 £15. Tel. 091-262 7764.

M128, manuals and large quantity of software £250, M512 board with mouse, manual and GEM software £95, Quantity of PC Public Domain software - free with 512 board, 3 Master ROM cartridges and quantity of ROMs £20, Sanyo 14" colour monitor £100, BBC B

issue 4 with some DFS chips £100, BBC Advance Reference manuals £10, BCPL Language ROM with manuals, boxed £15, Newbury Data 8820 Wide Carriage printer & extra head, ribbons, ideal for large quantities of listings and spreadsheet printing £85, 30 Hour Basic (book) £2, Holed out extra courses Vol.1 (disc) £6, P.I.A.S. 14 (disc) £6, Marlis Golf Database (disc) £6, Arcade Game Creator (disc) £4, Combat Lynx (tape) £1, Meteors (tape) £1, Philosophers Quest (tape) £1. All above in excellent condition and price negotiable. Tel. (0752) 896077.

M128 £200, Interword, Viewstore, Spellmaster £15 each, Master reference manuals £15 pair, Philips green screen monitor £25, twin disc drives £15, Panasonic KXP1080 printer £70. Tel. (0705) 594845.

Griffen Interface for BBC Micro also Analogue to Digital pack, DCP Expansion bus, Octal buffer board, Experimental manual, leads, instructions and software, as new £50 the lot. Tel. (0480) 453381 after 6pm.

Viglen external ROM cartridge system £5, & cartridges (12x£2), Graphics extension (CC) £10, Printmaster (CC) £10, Logotron Logo £30, Exmon II £15, Superart & mouse £35, Pagemaker £20, ESM Screenprint £10, BEEBUG Toolkit £10, Logotron Pendown £30, Integrex Screendump £5, Printwise (disc) £15, Quicicalc (disc) £5, Masterfile II (disc) £10. All the above have manuals/instructions, owner upgrading. Tel. (0234) 708463.

Comprehensive M128 system, co-processor, twin Opus 5.25" 40/80 drives, Cub monitor, NEC Printer, Acorn Teletext, ROM and Disc software, manuals, books and BEEBUG magazines. Silly to split as super Xmas gift for only £350. Tel. (0305) 848437.

BBC B (issue 7) 52k with Aries B20 and ROM expansion board £150 o.n.o. Tel. (0258) 452873 after 5pm.

BEEBUG Toolkit plus ROM for BBC B complete with instruction manual £10. Tel. (0246) 274436.

BBC B issue 7 with sideways RAM & CARE Low profile ROM module, Midwich 40T and Watford d/s 40/80 drives, Philips amber monitor, Medusa driver for Citizen printer, View, Viewsheet, Viewstore, Viewspell, BEEBUG Sleuth, Edikit and Toolkit, Educational software e.g. Key, Inform, Mappit, complete set of BEEBUG magazines - to April 92 (current subscription), Acorn User Jan 85 - Dec 90, various books including Advanced User Guide, large number of 5.25" discs, mostly with educational software (Geography), Citizen 120D printer and spare ribbons, BBC B ROMs, disc drives and monitor £250. Other items negotiable. Tel. (0472) 83441.

512 board for Master with Dr DOS 2.1 £90, Z80 second processor £65, ATPL board with battery backup £30, Softlife numeric keypad £15, Technomatic EPROM blower £30, Microvitac colour monitor £95, Model B with 1770 and Acorn speech system £150, also loads of software available. Tel. (0372) 740678.

RX80 printer good condition hardly used £85 o.n.o. Tel. (0277) 632244 extn 254.

Amstrad Laptop 20Mb & Canon Bubblejet printer, Wordstar Express & Supercalc, with guarantee, boxed unused, unwanted gift. 1st offer at 3/4 list price secures. Tel. (0590) 673929.

WANTED: Hardware project books, also books on the analogue and user ports. Tel. (0673) 843572 and ask for Mark.

WANTED: Comal ROM & Pacal ROM, Generator, Manual for Master, tuner for Philips CM8833 and BBC/Master books. Tel. (0383) 720656.

M128 £200, M512 £450, Philips monitor £60, Cumana DD £125, Teletext adaptor £50, Juki 6100 printer £150. Tel. 071-221 8450 eves.

M128, Acorn colour monitor, 40T dual disc drives £350, Overview £25, Master ROM £20, Viewstore £20, Morley Teletext adaptor £40, manuals & software, bits and pieces. Tel. (0533) 812904.

WANTED: ATPL ROM board, preferably with battery backup for BBC B & 128. Tel. 081-555 9069.

WANTED: Gilsofts The Quill (disc) or ALPS Adventure Creator (40T for BBC B) and The Advanced User Guide. Tel. (0652) 53981 after 6pm.

BBC B & M128 software and hardware for £1.50, please write enclosing sae for comprehensive list to: Mr J Ribbons, 30 Overcote Lane, Needing Forth, Huntingdon, Cambs, PE17 3TU.

M128, latest OS ROM, lots of extras £300, Philips CM8833 colour monitor £160, DSDD 80T DD including PSU £60, Acorn genuine 21Mb hard drive £250, DOS Plus 512 board including Essentials 1Mb upgrade, Acorn mouse, original discs & manuals £260, twin ROM cartridge £5, M128 double housing plyth £10, offers invited. BEEBUG binders Vols.1-9 £70 Master reference manuals 1&2 £10, M512 Technical & User guides including discs £20, Elite & Exile £5 each, Problem Solver & lots of essential software discs £5 each, Tulls mouse driver & compactor £5 each, genuine reason for sale, all above in mint condition. Tel. (0326) 240734 after 6pm.

PRES Plus1 and Plus3 for Acorn Electron, includes all manuals and Welcome disc, View cartridge and printer driver included, boxed as new £80 & carriage. **WANTED:** on 3.5" disc for Compact, Strike Force Harrier and/or Red Arrows. Tel. (0482) 46812.

BBC B with sideways RAM, Acorn 1770 DFS, Opus double 40/80 DD, Brother HR5 printer, mono monitor, Publisher DTP, View, Mini Office II (tape), other software & games, tapes & discs, books etc. boxes and manuals included £200. Tel. (0760) 23244 eves.

AMX mouse & Super Art ROM £35, Fleet Street Editor £20, Fleet St. Fonts 'n' Graphics and Admin Xtra packs £7.50 each, Advanced disc investigator ROM £15, WE Lightpen & software £7.50, AMX MAX Desktop & AMX XAM Quiz software £10 each, WE Colour Art Disc £2.50, also Games; discs (inc. Aviator, Firetrack) £3 each, tape completions £3 each and other tapes up to £1 each, books (inc. Mastering Assembly Code) up to £5 each, all prices are exc. p&p and are o.n.o. Tel. (0272) 684209.

WANTED: for BBC Compact, Mertec Compact Companion or a PearTree HC1 2MHz Bus interface, Morley

Teletext adaptor, RS 232 interface IC's. Tel. (0452) 830146 eves or (0452) 305752 day.

INFORMATION WANTED: BBC Speech system guide speaks of extending Human Speech beyond "Word PHROM A", did other PHROMs become available or any plug in cartridges? Is there any (Human) speech system for BBC B suitable for use in teaching English? Help desperately needed. Tel. (0533) 376180.

Archimedes 310m colour system, second disc drive, Backplane, Epson RX80 printer, software and 4 vols. of RISC User magazine & disc. Offers? Tel. (0526) 44559.

Shadow, Sideways RAM, ROM board, Solidisk 4Mb 32k, all on one board, 32k RAM plus extra 32k RAM, 5 ROM sockets, Manager plus utility software on disc (ROMable), plus extra software, compatible with View series and Inter series ROMs, boxed with manual. Perfect order £25. Tel. (0276) 65512.

BEEBUG Vols. 1-9 bound complete with indexes, £10 each and BEEBUG magazine discs Vols. 6-9 £15 each plus carriage. Tel. (0223) 870750 eves.

Cambridge co-processor version 1.4, 1Mb RAM, Acorn 10Mb hard disc with Basic, Fortran 77, IsoPascal, Lisp, C and 32016 Assembler can run with BBC B or Master, complete set of manuals £400 the lot. Tel. (0946) 723275.

Is there anybody out there who can sell me the Mertec Expansion box which is needed for the Wapping Editor to go with the Master Compact. Contact: Mr G Bjorgolfsson, Kirkjubraut 25, Akranes, ICELAND.

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. (0438) 573688.

512/1024 (PC+) board, mouse, Gem software, DOS 2.1, Shibumi Problem Solver 1.2, 512 User Guide, Dabhand 512 User & Technical Guides & discs £170, Morley master "AA" ROM board £25, Morley 1MBit RAM disc unused £100, Genie cartridge £20, BEEBUG Vols. 1-6, Disc User mag and discs 12 issues, Fast Access - 12 discs and cards and binder, A&B Computing up to Oct 89, Micro User up to April 90, discs from May 89, Acorn User up to August 90, some missing, offers. Buyer collects or pays postage. Tel. (0506) 883071 after 6pm.

BBC B recently upgraded with DFS, excellent condition, sensible offers around £125. Tel. (0277) 654332.

A3000 Learning Curve, mint condition including 2Mb RAM, colour monitor, Star LC10 printer, Genesis, 1st Word Plus, & lots of educational software £700 o.n.o. Tel. (0386) 553841 eves.

M128, Opus colour monitor £70, Weltrac 40/80T twin disc drive £95, Juki 6100 daisywheel printer £115, or bargain price for the lot including dustcover, manuals, joystick £425 o.n.o. Tel. (0483) 63561.

Dual 80T disc drive £60 o.n.o. Tel. (0273) 832548 eves.

Computer Village, ROM/RAM expansion board, CVx16-2, battery backed and inbuilt read/write protection, 14 ROMs & 2 8k sockets of RAM, 9 configurations of board £25 plus p&p. **WE 32k RAM card for BBC micro & manual** £20 plus p&p. **Acorn disc drive 40T 100k single sided** £25 plus p&p. Tel. 081-529 2949.

Interword complete £25, Opus Challenger 1Mb RAM disc & Opus DDOS+ 40/80T disc drive (needs attention) including ROM and manual £45 + postage. Tel. (0829) 270176.

WANTED: Z88 computer c/w mains adaptor, possibly with case and links for BBC or PC. Tel. (0703) 842646 (home) or (0703) 662265 (work).

BBC B and Microvitec 452 Cub monitor, also Cumana disc drive, all connecting cables in excellent condition £550 o.n.o. **BEEBUG magazines from Vol.1-10** 93 magazines in all £75. Tel. 081-346 7228 wk/days after 4pm or all day wk/ends.

M128, DD, printer, carrying case, ROM cartridges, software including Interword, Intergraph, Spreadsheet, manuals, magazines, discs £275 or will separate. Tel. (0462) 676761.

Watford ROM/RAM board for BBC B plus utilities disc with printer buffer also BEEBUG Spellcheck III ROM with utilities disc and Wordwise Plus ROM, all handbooks available, offers over £25 for the lot. Tel. 041-334 5676.

Tape based software - Includes games; 737 Simulator and Educational Format, also Advanced User Guide & 30Hr Basic manual, BEEBUG magazines from April 82 until Sept 91 £50 the lot or will split. **Mini Office II 80T £10, Board games, Arcade games** £5 (both discs). Write to: Mr W Plowman, Springfield, Westhay Road, Meare, Glastonbury, Somerset, BA6 9TC.

We like to accommodate as many reader's ads as possible each month, so please keep your ad as short as possible.

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

£27.50
£41.50
£50.50
£55.50
£59.50

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) 268328 ISSN - 0263 - 7561

Magazine Disc

October 1991 DISC CONTENTS

AN AUTOMATIC SAVE UTILITY - an extremely useful utility for anybody who types in long listings. It provides an automatic save facility which works in any selected screen mode.

MASTER MOS DFS BUGS - a short listing which offers a solution for the OSWORD & 7D/7E bug in DFS 2.45 as supplied in the new Master ROM, and a solution to the problem of switching from ADFS to DFS.

TEXBASE : A FREE FORMAT INFORMATION SYSTEM (PART 2) - the two programs from last month for entering and storing text, and the new Search routine adding the search options to the database.

WORDWISE USER'S NOTEBOOK: WORDWISE PLUS TELETEXT EFFECTS - four segment programs for users of Wordwise Plus which provide security on your screen, colour for displays, positioning of text and special effects.

BEEBUG WORKSHOP: SIMULATION MODELLING - a program which demonstrates the modelling of a simple queuing system.

ADFS DESKTOP ENHANCEMENTS - an enhanced version of the useful 'front-end' ADFS Desktop utility, published in Vol.8 No.10 offering some new handy features. The complete, updated system comprising two programs is included.

DOUBLE SIZE CHARACTERS IN MODE 0 - two procedures for creating double size characters in 80 column modes - the first one is in Basic and the second is a faster Assembler version.

NUMERICAL NOTATIONS - this program contains procedures for improving the presentation of floating point numerical output.

BEEBUG FUNCTION/PROCEDURE LIBRARY (6) - this month we add a sound library and a procedure for plotting dotted lines at any angle on the screen.

HEXAGONS - an entertaining and brain taxing game where you need to rearrange seven regular hexagons so that the colours of the different segments match.

MAGSCAN DATA - bibliography for this issue

Duration: 240

Arrival times

Mean: 5

Standard deviation: 1.5

Service times

Mean: 6

Standard deviation: 1

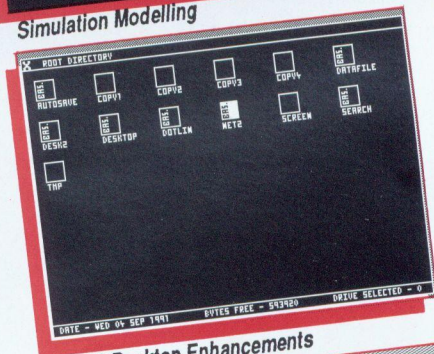
Clock = 11

queue = 0

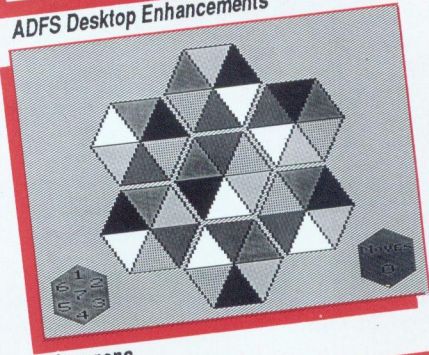
Clock = 12

queue = 1

Simulation Modelling



ADFS Desktop Enhancements



Hexagons

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)	UK ONLY	OVERSEAS
12 months (10 issues)	£25.50	£30.00
	£50.00	£56.00

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

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

Special Offers to BEEBUG Members October 1991

BEEBUG'S OWN SOFTWARE

Code	Product	Members Price	inc Vat	Code	Product	Members Price	inc Vat
1407a	ASTAAD3 - 5" Disc (DFS)	5.95		PAG1a	Arcade Games (5.25" 40/80T)	5.95	
1408a	ASTAAD3 - 3.5" Disc (ADFS)	5.95		PAG2a	Arcade Games (3.5")	5.95	
1404a	Beebug Applics I - 5" Disc	4.00		PBG1a	Board Games (5.25" 40/80T)	5.95	
1409a	Beebug Applics I - 3.5" Disc	4.00		PBG2a	Board Games (3.5")	5.95	
1411a	Beebug Applics II - 5" Disc	4.00		1600a	Beebug magazine disc	4.75	
1412a	Beebug Applics II - 3.5" Disc	4.00		0077b	C - Stand Alone Generator	14.56	
1405a	Beebug Utilities - 5" Disc	4.00		0081b	Masterfile ADFS M128 80 T	16.86	
1413a	Beebug Utilities - 3.5" Disc	4.00		0024b	Masterfile DFS 40 T	16.86	
1450a	EdiKit 40/80 Track	5.75		0025b	Masterfile DFS 80 T	16.86	
1451a	EdiKit EPROM	7.75		0074b	Beebug C 40 Track	45.21	
1452a	EdiKit 3.5"	5.75		0075b	Beebug C 80 Track	45.21	
0005b	Magscan Vol.1 - 8 40 Track	9.95		0084b	Command	29.88	
0006b	Magscan Vol.1 - 8 80 Track	9.95		0073b	Command(Hayes compatible)	29.88	
1457b	Magscan Vol.1 - 8 3.5" ADFS	9.95		0053b	Dumpmaster II	23.76	
0011a	Magscan Update 40 track	4.75		0004b	Exmorn II	24.52	
0010a	Magscan Update 80 track	4.75		0087b	Master ROM	29.88	
1458a	Magscan Update 3.5" ADFS	4.75		1421b	Beebug Binder	4.20	

0% Finance

A3000 Learning Curve

For a limited period we are again able to offer 0% finance for the Acorn Learning Curve Package

As well as the A3000 computer, this package includes the **1st Word Plus** word processor, **Genesis** educational database and the **PC Emulator** to run DOS programs.

Also included is a **video** on how to set-up the computer and information on the National Curriculum.

**Please phone for written details:
0727 40303**

OTHER MEMBERS' OFFERS

These offers are available for a limited period only, while stocks last. Orders are dispatched on a first come first served basis. To order phone 0727 40303.

COMPLETE BUSINESS PACK FOR THE MASTER 128

OverView

offers a database, spellchecker, index creation utility and a business graphics package

Members Offer **£22.35** inc VAT + p&p

Normal price **£93.60** inc. VAT

OverView from Acorn combines all View family programs into one package, and complements View and ViewSheet supplied with the Master 128.

- ◆ OverView adds **ViewStore**, **ViewSpell**, **ViewPlot**, **ViewIndex** and the **Printer Driver Generator**.
- ◆ The ROMs are supplied in an Acorn ROM cartridge.
- ◆ Switch between packages without having to save files.
- ◆ The postage cost reflects the size/weight of the package.

Stock code 1020E

Post & Packing	UK	Europe	Americas, Africa Middle East	Elsewhere
a	£ 1.00	£ 1.60	£ 2.40	£ 2.60
b	£ 2.00	£ 3.00	£ 5.00	£ 5.50
c	£ 3.10	£ 6.50	£10.50	£11.50
d	£ 3.60	£15.50	£24.50	£25.50
e	£ 6.00	£15.50	£24.50	£25.50